On Security in Publish/Subscribe Services: A Survey

Christian Esposito and Mario Ciampi

***Abstract*—Publish/subscribe services have encountered considerable success in the building of modern large-scale mission-critical systems. Such systems are characterized by several non-functional requirements, among which security plays a pivotal role due to the emergence of numerous cyber attacks targeting most mission-critical systems. This requires that the adopted publish/subscribe services have to be equipped with the proper means to protect the exchanged data, to preserve their correct behavior and to face possible attack scenarios. Although significant efforts have been made in this field, many issues are still open. This paper includes an introduction to the principles of securing event notification, and an analysis of the relevant state-of-the-art by both surveying the academic literature over the period 1998–2014 on secure publish/subscribe services and overviewing the current standards for the marketed products. Next, it presents the main challenges that are still unresolved and are worthy of further attention in future research efforts.**

***Index Terms*—Security, multicast, publish/subscribe services.**

I. Introduction

PUBLISH/SUBSCRIBE services [1] have proved to be a suitable solution for the building of large-scale communication infrastructures thanks to their intrinsic asynchronous communication and decoupling properties. In fact, these properties remove the need to establish explicitly all the dependencies among the interacting entities, so as to make the resulting communication infrastructure more scalable, flexible, and maintainable. For this reason, publish/subscribe services are currently used as the main building component in several past and on-going industrial projects to realize innovative large-scale infrastructures in different application domains (such as in healthcare applications [2], air traffic management [3], or manufacturing production [4]). In fact, widely-known standards for publish/subscribe services, e.g., OMG Data Distribution Service (DDS) [5] and Java Messaging Service (JMS) [6], represent the key technology in several examples, such as the ones shown in Table I.

Most of these large-scale infrastructures can be considered as practical examples of a Critical Infrastructure (CI), which represents the pivotal assets and resources for the correct functioning of a society and economy, and constitutes the backbone of a given nation's economy, security, and health. The peculiarity of a CI is that its damages or malfunctions, intentionally or accidentally caused, may have a serious impact in terms of human suffering or the loss of human life, the damage, or even destruction, of economic structures, and environmental degradation. Therefore, CI Protection assumes a vital role in assuring a nation's security, safety, efficient governance, sound economy, and high level of public confidence. This is proved by the proliferation of numerous efforts by governments all over the world to clearly define national CIs, to identify their possible vulnerabilities and to determine effective methods of protection. The adopted middleware solution used to interconnect and integrate the components within a CI plays a significant role in the correct behavior of such an infrastructure. Accordingly, stringent non functional requirements are placed upon such middleware, and special attention is paid to security.

The strong interconnection among its parts as well as the opposite trend of deploying installations in a more decentralized way make it necessary for CIs to have multi-point cooperative communication patterns and a data-centric approach for data dissemination, so as to handle their large scale. For this reason, publish/subscribe services are considered a winning choice to realize data dissemination within a CI, and, accordingly, we have witnessed their successful adoption in several attempts to develop innovative large-scale CIs, such as those in Table I. However, publish/subscribe services do not only have to satisfy the technical challenges of multi-point cooperation and data-centric communication, but also non-functional requirements, especially security. Therefore, there is the mandatory need of secure publish/subscribe services able to provide a suitable level of protection against possible attacks.

Security in publish/subscribe services is a recent topic in the research community of this kind of middleware, since such a community was previously more focused on scalability and expressiveness. However, as soon as publish/subscribe services were progressively adopted in important industrial projects, such as those mentioned in Table I, with stringent security issues to be faced, several approaches have been proposed and industrial standardization bodies have issued recommendations related to this topic. However, at the moment there is no agreement on which attributes should compose the security property for publish/subscribe services and, thus, several different techniques have been proposed, each tailored on a particular aspect of the overall security concept. The goal of this survey is to clearly define a security model for publish/subscribe services and to investigate the pros and cons of the available academic solutions so as to highlight what is still lacking in this topic in the current literature. This is achieved by surveying the literature over the period 1998–2014 on security in publish/subscribe services by comprehensively reviewing the developments in this area and covering the main available solutions. Moreover, several industrial standards are presented so as to point out

Manuscript received October 14, 2013; revised April 16, 2014 and August 7, 2014; accepted October 17, 2014. Date of publication November 5, 2014; date of current version May 19, 2015.

The authors are with the Institute of High Performance Computing and Networking (ICAR) at the National Research Council (CNR), 80131 Napoli, Italy (e-mail: christian.esposito@na.icar.cnr.it).

Digital Object Identifier 10.1109/COMST.2014.2364616

TABLE I
USE CASES FOR DDS AND JMS, TAKEN FROM THE CUSTOMERS OF FUSESOURCE [7], RTI DDS [8], AND OPENSPLICE DDS [9]

User	Application	Standard	Description
CERN	Operational grid activities (monitoring systems) of the Large Hadron Collider (LHC)	JMS	Integrating over 100,000 machines in 20 different countries so as to form a grid for processing operational monitoring data from the LHC and other scientific instruments of CERN
FAA	Transformation of the US National Airspace System (NAS)	JMS	Interconnecting the existing FAA systems, industry, and airline partners so as to share information and handle increased capacity and future demand
Lynden	Track freight in real time	JMS	Keeping warehouses, transportation specialists, as well as the customers up to date at all times
Sabre Holdings	Handling travel transactions among agencies	JMS	Realizing a platform for connecting travel consumers to travel suppliers
Apex Network	Vehicle recovery and roadside assistance	JMS	Creating a network for the automotive industry for connecting motoring organizations, recovery operators and industry suppliers
Health departments of some Italian regions	Health Information System	JMS	Designing and implementing a health information system to exchange electronic health records among health structures in different regions.
EuroControl	Novel Air Traffic Management (ATM) framework in Europe	DDS	Integration of all the ATM systems within Europe
City of Tokyo	Highway Traffic Monitoring	DDS	Interconnecting roadway sensors and roadside kiosks to a centralized control center so as to deliver constant updates to kiosks and to gather traffic condition data from sensors
Volkswagen	Advanced driver assistance	DDS	Integrating complex environment perception and control systems into the car
Amsterdam Metro	Rail-Based Traffic Management/Monitoring	DDS	Real-Time monitoring and control of every single element in the railway system
Atlantida	Unmanned Air System (UAS)	DDS	Allowing communications among unmanned aircrafts, control systems, and other related support equipment, so as to automate air vehicle operations
U.S. Department of Defense	Future Combat Systems	DDS	Using information systems for the Net-Centric Operations and Warfare (NCOW) environment so as to enable enhanced joint connectivity and situational awareness in battlefields
Grand Coulee Dam	Power Plant monitoring and control	DDS	Interconnecting 40.000 SCADA systems controlling the 30 generators of the dam and the transmission switchyard
U.S. Department of Energy	Smart Grids	DDS	Implementing a wide-area network capable of monitoring and controlling an entire power grid
City of Nice	Smart Cities	DDS	Realizing an information backbone to ensure a smart mobility and environmental quality in the city

what the current commercial or open-source products are able to provide in terms of security.

This survey is targeted for the generalist in the fields of middleware and networking, and not only limited to experts within the context of publish/subscribe services. For this reason, Section II provides some background to support our discussion by presenting the key concepts of publish/subscribe services, group communication systems, and the main security methods and concepts. Next, Section III goes into detail about security in publish/subscribe services by listing the relevant properties, defining a threat model and a solution space to protect such services from malicious adversaries. On the one hand, Section IV provides an overview of the available academic solutions so as to be of help for researchers looking for the open issues still unresolved within the current literature. On the other hand, Section V describes the main industrial standards for publish/subscribe services by highlighting the provided security means so as to be of help for industrial practitioners wanting to select the best standard for their needs. We conclude the paper with Section VI by summarizing our findings and providing some final remarks.

II. BACKGROUND

A. Publish/Subscribe Services

Publish/subscribe services [1], [10] represent a middleware solution where messages are delivered to interested destinations in a data-centric manner, i.e., without indicating the address of such destinations but based on some attributes of the messages, and an anonymous interaction style, i.e., both senders and receivers are not aware of their respective identities. Such services belong to the so-called Distributed Event-Based Systems (DEBS), [11], [12]. Their name results from the fact that at the beginning they only represented services offered by mature and standard Client/Server middleware platforms, such as Common Object Request Broker Architecture (CORBA) providing event notification with two specifications, i.e., CORBA Event Service (CORBA-ES) [13] and its evolution CORBA Notification Service (CORBANS) [202], or Web Services Notification (WSN) [14] introducing event-based communication within Web Services. Later on, they evolved into proper middleware architectures, such as the DDS [5], JMS [6], or the Advanced Message Queuing Protocol (AMQP) specification [15], but the term "Service" routinely remained in their name. In such middleware solutions, information is denoted as an *event*, which is a detectable condition in the application, e.g., any change in its internal state, and the act of delivering an event is indicated as a *notification*. Such architectures are based on the popular Object-Oriented design pattern called *Subject-Observer* [16]: an object, called an *Observer*, is automatically notified and updated about changes that occur in the state of another object, called a *Subject*. The observer has to inform the subject which events it is interested in by means of a so-called *Subscription*. Then, if the subject detects the occurrence of one of these events, it notifies the observer by invoking one of its methods. Such a design pattern presents the following benefits: (*i*) minimal coupling between subjects and observers, (*ii*) support to multicast communication of state changes, and (*iii*) filtering of notifications based on observer interest. However, it also has the drawback of each subject having to maintain a list of observers, with their related interests, and to call them as necessary. To improve decoupling, and thus scalability, the Subject-Observer pattern has been combined with the *Mediator* pattern [16], giving rise to the *Event Notification* pattern [17]: the process of managing observers and notifying them is not locally placed with the subjects, but centralized at a third mediating entity, sometimes referred to as the *Notification Service*. Strong decoupling among applications is obtained by using *Implicit Invocations* [18] of observer methods when events occur in the subject. This pattern, combined with the *Reactor* pattern, which dispatches incoming notifications to several handlers, and the *Proactor* pattern [19], which uses asynchronous operations, has made possible a novel interaction model that has been formalized in the *Publisher/Subscriber* pattern [20], depicted in Fig. 1.

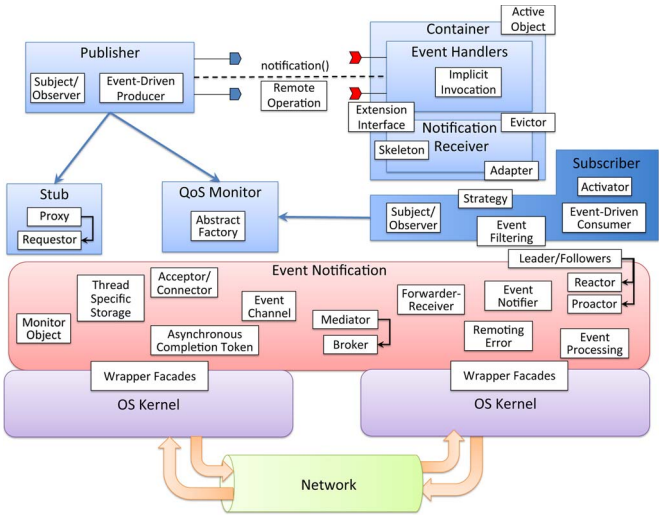

Fig. 1. Publisher/Subscriber Pattern described with the Pattern Language taken from [19], [21], and [22].

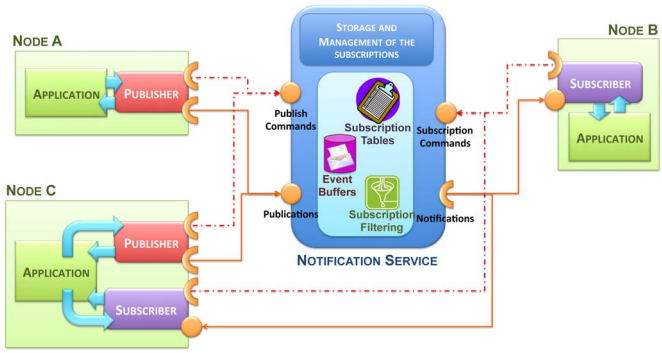

Fig. 2. Architectural Overview of a Publisher/Subscribe Service (the solid lines represent the event flows from publishers to subscribers, while the broken lines indicate the commands that the Notification Service receives from publishers and subscribers).

Such a pattern defines the architectural scheme that characterizes a typical publish/subscribe service and is composed of three main entities: (*i*) a *publisher* that disseminates the generated notifications of the events occurring in the application; (*ii*) a *subscriber* that receives the notifications of its interest and passes them to the application; and (*iii*) the *notification service* that interconnects publishers and subscribers, and has the function of storing incoming notifications and of routing them to interested subscribers. As illustrated in Fig. 2, the nodes constituting the service may contain one (such as the nodes A and B in the figure) or both (such as the node C in the figure) of the two key client components, i.e., the publisher and the subscriber, apart from the component encapsulating their business logic (indicated as *application* in the figure). Moreover, the subscriber's interest in certain notifications is specified by means of *subscriptions*, which are passed by subscribers to the notification service. At the reception of a new notification, the notification service verifies which of its subscriptions are satisfied by the received notification, and forwards it to the subscribers that have registered the satisfied subscription. Therefore, between the notification service and the subscribers we can have two kinds of interactions. In the first (indicated in the figure as Subscription Commands), the subscriber registers, or unregisters, subscriptions on notifications of interest, while in the second the notification service passes all the notifications that verify the registered subscriptions. In addition, between the publishers and the notification services we can have two kinds of interactions. In the first, a publisher sends new notifications, while in the second, a publisher advertises which kind of notifications it is about to send or to stop publishing (indicated in the figure as Publish Commands). The notification service allows a decoupling of the production and consumption of events, which has resulted in an increase of scalability [23] since all explicit dependencies have been removed [1].

The peculiarity of publish/subscribe services is the way of expressing subscriptions:

1) *Channel-based* [13]: the notification service holds a set of named channels, i.e., a distributed object univocally identified with a *channel_id*, to which (*i*) publishers have to connect so as to forward their notifications, and (*ii*) subscribers have to listen so as to receive their notifications of interest.
2) *Topic-based* [24]: publishers tag outgoing notifications with a topic string, while subscribers use string matching to detect their events of interest.
3) *Type-based* [25]: subscribers express their interest only in events that present a certain structure or event type.
4) *Content-based* [26]: subscribers express complex predicates on the event content, and events are delivered only if they satisfy such predicates.

A publish/subscribe service can present one of such subscribing approaches, such as the CORBA-ES or JMS, or also a combination of them, such as DDS that combines type-based and topic-based subscriptions or the CORBA-NS that jointly uses channel-based and type-based subscription.

A set of protocols is adopted by publish/subscribe services so as to realize the distribution of notifications from publishers to the interested subscribers. In fact, such services can be realized by exploiting the communication capabilities of traditional unicast transport-level protocols, such as UDP or TCP; but they can also use transport- or network-level multicast and/or broadcast primitives for this objective. Within the current literature, several possible solutions have been proposed [27]–[29], which are part of the research topic of group communication. The next subsection is devoted to a presentation of the theory of group communication and how it is related to publish/subscribe services.

B. Group Communication

Group Communication [29] is the ability of a group of software components or applications to exchange data in a one-to-many or many-to-many fashion. The means to realize such a communication is structured in two different steps: (*i*) group definition and (*ii*) multicast. The first step defines which entities belong to a given group and assigns a unique identifier to a given group. Multicast consists of the means for exchanging data within a group characterized by a given identifier. With respect to this, we can indicate a group as "closed" or "open" if the data source is obliged, or not, to be a member of the group

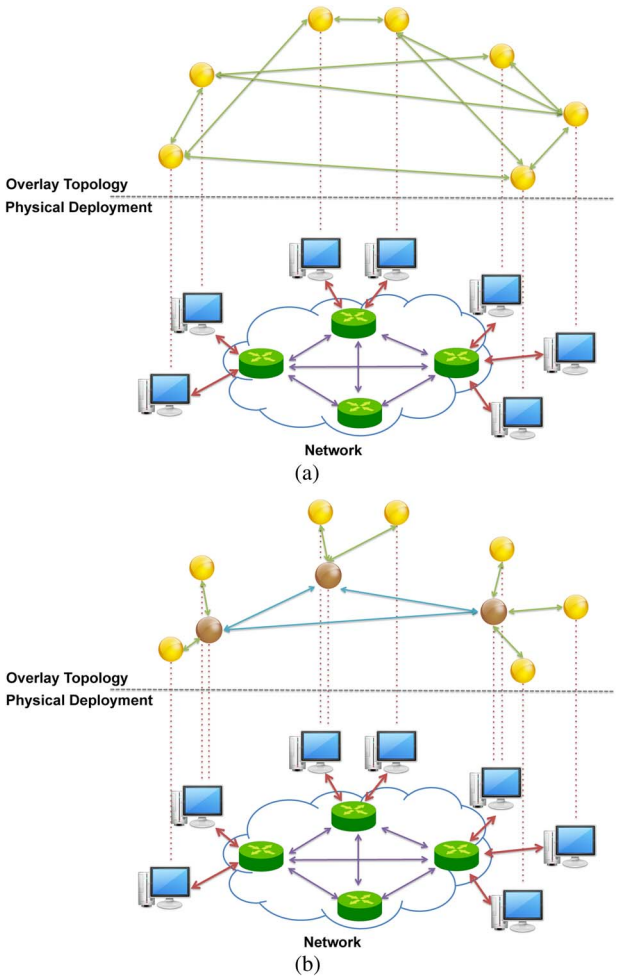

Fig. 3. Two possible classes of overlay-based multicast.

within which its generated data are disseminated. Multicast has been an active research topic in the last decades [30]–[32], and the designed multicast protocols are traditionally grouped in two main classes [33]: Transport-Level Multicast (TLM) and Application-Level Multicast (ALM). As the name suggests, TLM protocols realize the multi-point data delivery at the network layer of the ISO/OSI stack by adopting IP Multicast [34], [35]. Group definition is based only on the knowledge of a certain multicast address to be used to send/receive messages towards/from the group. IP Multicast provides an efficient use of the network resources and allows the developers to easily realize a multicast service. However, there are several flaws that have limited the adoption of TLM, especially in large-scale settings [36].

ALM implements the multicast at the application layer [32], [33] by using proper transport-level protocols, such as TCP or UDP. The members of a group are interconnected using an overlay network, where the communications over the overlay links are conveyed by TCP or UDP. The duty of replicating the messages to be dispatched over several distinct outgoing links is carried out by the group members, such as software components or applications, rather than routers. The adoption of an overlay allows ALM to avoid both the deployment issues and the scaling limitations that affect IP Multicast over wide area networks. However, ALM is not able to achieve the same efficiency as TLM in terms of network resource usage. A taxonomy of the ALM solutions realized so far over the years can be structured into two distinct groups:

- *Infrastructure-less overlays*, shown in Fig. 3(a)—the overlay vertices are directly the group members in accordance with a peer-to-peer topology. The benefit of this solution is a low performance degradation since each member can potentially communicate in a direct manner with all the others. However, this imposes some scalability issues when managing large groups.
- *Infrastructure-based overlays*, depicted in Fig. 3(b)—the group members are not directly interconnected within the overlay, but one or more entities mediate among the group members. Such entities are based on the *Broker* design pattern [21], so they are named as brokers (but in the current literature they have been indicated also as rendezvous nodes, or dispatchers). The brokers contain the needed forwarding logic to collect incoming messages, to extract the groups to which such messages have to be disseminated, and to distribute them to all the members of the destination groups. The benefit of having mediators is that of relaxing the scalability issues of infrastructure-less overlays at the expense of a higher performance cost.

Regardless of how the overlay is supported, we can further distinguish between two different organizations [37]:

- *Tree-based structures*, with the nodes structured as a tree, where each group member can implicitly define its parent, from which it receives the incoming messages, and children, to which it dispatches the incoming messages;
- *Mesh-based structures*, with a less structured organization letting each group member employ a swarming delivery mechanism to a certain subset of peers.

Mesh-based approaches are more resilient to crashes and disconnections since they do not have a rigid structure and can more easily adapt to changing conditions. However, meshes are more complex to build and to maintain than trees, since the nodes need to have knowledge of some of the others, which is challenging to obtain in Internet-scale systems.

As above mentioned, a publish/subscribe service is generally built on top of a group communication service [38] so as to implement the needed notification service. In the case of channel-, topic-, and type-based publish/subscribe services the membership of a certain group is defined by the submissions, e.g., given a topic all the subscribers are interested in such a topic and all the publishers that produce events belonging to the given topic can be clustered within a group identified by this topic. On the contrary, content-based solutions do not present such static definition of groups, but the membership of a certain group rather than of another one is dynamically defined based on the content of the published events. In fact, for a concrete example, DDS is built on top of IP Multicast, while JMS and several academic prototypes, such as Siena [39] or Scribe [40], use an overlay-based group communication service, which can be organized in several different ways: some use Tree-based structures, while others Mesh-based structures. As a concrete example, Scribe [40] uses an infrastructure-less tree-based ALM built in a distributed manner

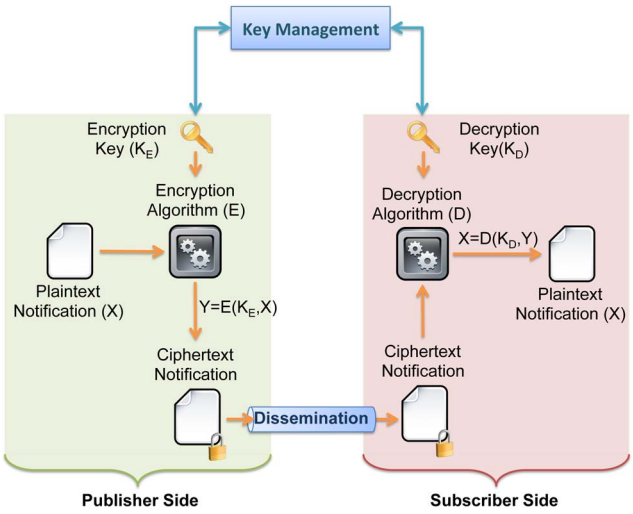

Fig. 4. A simple model of a notification encryption.

when a new node (i.e., a subscriber) wants to join a group. Specifically, it sends a *JOIN* request, which can reach a node already connected to the group or not. In the first case, the contacted node becomes the parent node within the tree of the new node. In the second case, the node passes the request to another one, until it finds a node joined to the group or the request reaches the root of the tree (i.e., a rendezvous node that collects publications from publishers and passes them to all the subscribers in the group). In this case, the new node becomes the child of the reached node within the tree. On the other hand, Siena [39] adopts an infrastructure-based mesh topology, by having its brokers organized in a peer-to-peer relationship and assuming a general undirected graph with the possibility of multiple paths between brokers. Siena assumes a peer sampling service or a configuration file for the discovery of all the brokers, and the possibility of a broker to choose another one and to ask to be connected to it.

C. Main Security Methods

Several methods can be adopted within the context of publish/subscribe services so as to provide certain security guarantees. We provide a brief description of them.

1) Encryption: It consists in the process of encoding the notifications to be exchanged, so as to turn them into artifacts unreadable without previously decoding them to recover the original data, as depicted in Fig. 4. The encoding process is carried out by using a certain algorithm with an encryption key as input, while the opposite runs another algorithm, giving as input a decryption key. The secure exchange of the adopted keys are exchanged among publishers and subscribers prevents unauthorized users from accessing the information contained in the notifications. From the literature, we can distinguish between symmetric or asymmetric schemes [41], respectively if a single shared key is used for both the encryption and decryption or two distinct keys are adopted. The second solution is also known as public-key cryptography, since the encryption key is publicly available, while its paired decryption key is kept private and cannot be derived from the public one. Most of the symmetric and asymmetric schemes are very complex and cannot be easily described, so we refer interested readers to [41], [42] for more details. Having two keys implies a simpler management of the used keys and the confidentiality of the exchanged information, since in symmetric schemes the publisher has to distribute its key to all the interested subscribers or to use a Key Distribution Center (KDC). On the contrary, in asymmetric schemes the privacy of notifications is assured thanks to the fact that the decryption key remains secret. However, the asynchronous schemes exhibit a higher computational overhead than symmetric ones [43], [44].

Traditional cryptography schemes are known to be deterministic: a given plaintext will always be transformed into a unique ciphertext. This may cause the leakage of partial information on the plaintext to an eavesdropper, the ease of performing a statistical analysis of the transmitted messages, or the correlation of intercepted ciphertexts with observed actions [45]. To deal with such problems, in the literature we can find some probabilistic cryptography schemes, such as the *Pailler homomorphic cryptosystem* [46], *where* a given plaintext can be transformed into more than one ciphertexts, and the ciphertexts to be used are randomly selected. Despite being more robust to cryptanalysis and information leakage, probabilistic encryption is affected by the need to exchange larger messages and a consequent performance degradation [47].

A key is typically a random string, unrelated to the signer identity; therefore, a certificate authority is needed. The use of certificates may cause overheads and inefficiency, which can be resolved with *Identity-based crypto systems* [48], which have a practical implementation with Pairing-Based Cryptography (PBC) [49]. Specifically, the public key of a user is easily computable from the user identity by means of bilinear pairings [50], and without requiring a certificate authority, but a key manager is present to assign private keys to the corresponding public ones. Due to the involvement of a manager, identity-based crypto systems are affected by the key escrow problem, which makes such systems less secure. On the other hand, PBC reduces the number of keys to be managed. If the user identity is expressed by a set of attributes, we can make the decryption possible only if at least k of the attributes of the user key matches the attributes of the ciphertext. Such a solution is known as Attribute-Based Encryption (ABE) [51], and implements access control over encrypted data.

Publish/subscribe services have as their peculiarity the possibility of taking routing decisions based on the event content and available subscriptions. While encrypting notifications provides the clear benefits of protecting the exchanged events, it implies the impossibility of executing queries on the notification content if it has not been decrypted first. This will result in a reduced degree of security since brokers will have access to the notification content even if they are not trustworthy. Moreover, there may be increased performance degradation, caused by a decryption and a further encryption before and after evaluating the subscriptions. A better approach is to evaluate subscription on encrypted data, without revealing the plaintext. The naive solution [52] is to divide the notifications in words, each being separately encrypted, and to express subscriptions as queries with parameters encrypted with the same key of the notifications. However, this approach exhibits limited expressivity in the allowed subscriptions. More advanced solutions are

TABLE II
PROS AND CONS OF THE MAIN CRYPTO SYSTEMS

Scheme	Pros	Cons
Symmetric Schemes	Very fast encryption and decryption	Bother of agreement on the used key
Asymmetric Schemes	Simplier key management	Bad encryption and decryption performance
Pailler homomorphic system	Robust and able to compute arithmetic operations on encrypted data	Expansion issue, causing higher overhead and performance
PBC	Less keys to manage	Affected by the key escrow
ABE	Precise disclosure of encrypted data to only authorized users	Higher cyphertext size and number of keys to manage
SDE	Computing over encrypted data with no privacy violation	Limited scalability and worse performance

available in the literature, such as the mentioned homomorphic cryptosystems [53] or Searchable Data Encryption (SDE) [54], which allow making encrypted data searchable without previously decrypting them.

Table II summarizes the strengths and weaknesses of the described cryptographic schemes.

2) Digital Signature: Another technique for security enhancement is digital signature, associated by a publisher to every generated notification, as shown in Fig. 5. Specifically, a signature is basically a hash of the notification content encrypted by using the private key of the publisher. The subscriber can verify the integrity of the notification by hashing the content of the received message and comparing this value with the one obtained by decrypting the signature with the public key of the publisher. When a public key infrastructure is used to bind public keys with respective user identities, the identity of the signer is documented by a valid digital certificate, i.e., an artifact associating a user identity to his/her public key [55], provided and signed by a certificate authority. Such a certificate is inserted within the signed notification. A digital signature is confused sometimes with a Message Authentication Code (MAC) [41], which relies on symmetric cryptography to generate an encrypted tag attached to notifications to guarantee their authenticity and integrity. Despite the difference in the adopted cryptography scheme, digital signatures and MAC differ also because digital signatures also provide non-repudiation (by using timestamps and/or nonces to prove uniqueness and freshness of a notification [41], [56]), whereas MAC does not, and because both publisher and subscriber share the same key to manage MAC.

There are several ways to generate a signature, among which the most well known schemes, such as the *RSA signature scheme* [57], [58], are deterministic since the adopted hash function generates a unique fixed-size bit string from an arbitrary block of data. Such signature schemes are affected by the same security issues of deterministic encryption. Another kind of schemes, such as the *ElGamal signature scheme* [59] and Digital Signature Algorithm (DSA) [60], uses a collision-resistant hash function that allows a given notification to have more than one valid signature. As demonstrated in [42], probabilistic schemes are considered worse in terms of performance than the deterministic ones. Probabilistic schemes require smaller keys and produce smaller signatures than the deterministic schemes to achieve the same security degree [61].

The Merkle Signature Scheme (MSS) [62] is an interesting alternative to well established signature schemes, since it is not vulnerable when an effective quantum computer is built.

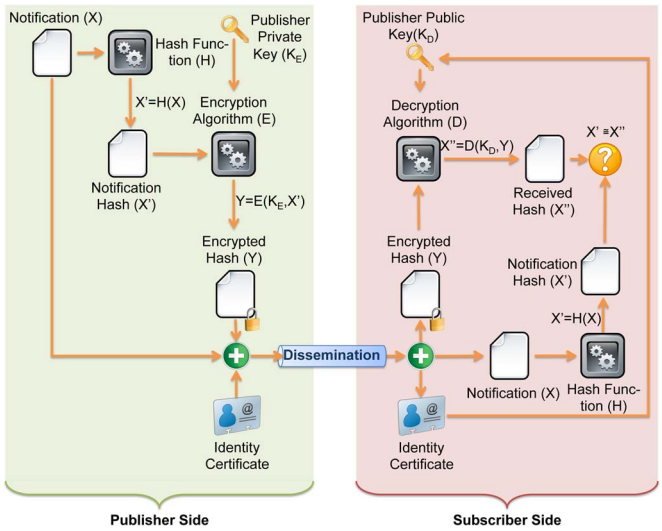

Fig. 5. A simple model of a notification signature.

TABLE III
PROS AND CONS OF THE SURVEYED SIGNATURE SCHEMES

Scheme	Pros	Cons
Deterministic schemes	Good performance and limited overhead	Privacy violation during verification, and security issues
Probabilistic schemes	Requiring smaller keys and generating smaller signatures	Privacy violation during verification, and bad performance
MSS	High security level	Privacy violation when verifying, long signatures and keys
Sanitizable Signature	No privacy violation during verification	Bad performance and high overhead

Reference [63] has shown that the MSS has comparable performance than the other schemes, but has longer signatures and private keys.

Signature verification is typically realized by accessing the notification content; however, this may compromise the provided privacy. Novel signature schemes have been proposed to enforce privacy in the signature verification, such as the *sanitizable signatures* [64], which can be verified even if a part of the message is not known, and the sensitive data hided during the sanitization process are not exposed when the signature is verified [65], [66]. Sanitizable signatures require keys with the same size as deterministic schemes, but exhibit worse performance.

The benefits and limits of the mentioned signature schemes are summarized in Table III. Such a table indicates that there is no silver bullet for digital signatures and each scheme has its own pros and cons. Moreover, such schemes are grouped within the class of *direct digital signatures*, since they involve only the communicating parties in the signature creation and verification. They assume that the subscriber knows the publisher's public key, whose security affects the validity of the signature scheme (i.e., the publisher and subscribers need to exchange the key securely avoiding it being stolen). It is possible to have the class of signature schemes named as *arbitrated digital signatures* [67], since they involves a trusted third party or arbiter, required to be trustworthy, that authenticates and validates the signed notifications without having access to their content. This allows a decoupling of publishers and subscribers from the border of key management, at the expense of an additional performance degradation.

Digital Signatures can be coupled with a freshness scheme [68], so as to protect against any possible reuse of the signature and to demonstrate that a given notification is secure and has not be replayed by an adversary [56]. Such a uniqueness can be enforced by several possible schemes:

- A timestamp can be put in each notification so as to consider it invalid if too old. Such a solution requires clock synchronization between the interacting entities, which is challenging in large-scale settings [69], [70].
- A notification can contain a sequence number, which is incremented after a new notification is published, and a received notification can be considered invalid if it contains a sequence number lower than the one in the latest received notification. Such a solution is vulnerable to message re-ordering, which can cause some messages to be erroneously discarded.
- The publisher can generate a nonce (i.e., a random number) and associate it to a notification, so as to let the subscriber rejecting those notifications having the same nonce seen in a previous notification. The issue with this solution is to find a manner to ensure nonces of being unique to each event.

Standards, such as the ISO/IEC 13888–3:2009 [56], and academic solutions, such as [71], encompass the possibility to have a Trusted Third-Party (TTP) to generate timestamps, sequence numbers or nonces for freshness check.

3) Key Management: Both cryptography and digital signatures require a wise management of the cryptographic keys, as indicated in Fig. 4. Such an issue is so important that a large amount of work on secure publish/subscribe services, and more in general on secure group communication, is focused on its realization. We can distinguish between two different approaches: a *link-by-link key management* and a *proxy-based key management*. In the first case, let us assume that the publisher P publishes a series of notifications, in which the subscriber S is interested. Then, P can locally generate the adopted keys, and physically derive, preferably in a secure manner, the one needed by S. Such an approach is widely used in unicast secure communication where the known solutions for encryption at the transport level, such as the Secure Socket Layer (SSL) or its evolution, namely Transport Layer Security (TLS), adopt a proper handshake protocol to negotiate the used encryption algorithms and to exchange the used cryptographic keys [41]. However, such a manual delivery is quite awkward for large-scale distributed systems. For a practical example, considering a system made of N applications using a symmetric encryption scheme, the required number of keys is $[N \cdot (N-1)]/2$, which need to be distributed and managed. This causes a huge overhead and compromises the scalability of the key management. For a more scalable and efficient solution, a *proxy-based key management* is adopted, where the responsibility of distributing, and even generating, the cryptographic keys is passed to a third-trusted KDC, which can be centralized or consist in a federation of several centers. Such a solution reduces the overhead for key management.

Despite simplifying the problem of distributing the used keys, the use of one or more KDC is not optimal since it is difficult to trust them. A different solution is to adopt mediating entities conducting proxy re-encryption schemes [72]: a semi-trusted proxy, called Proxy Security and Accounting Service (PSAS), transforms a message encoded with a given key into a message encoded with another key, without leaking the conveyed data. Such a technique removes the requirement to have a publisher and the interested subscribers aware of each other and agreed on the keys to be used. A concrete example of such an approach is provided by the proxy re-encryption scheme of Jakobsson [73]. The major limitation of this solution is the assumption that the proxy is honest, making it vulnerable to corrupted proxies. Jacobsson [73] deals with this problem by implementing the proxy with multiple servers, each of them performing partial re-encryption. However, this is not collusion-safe, i.e., the multiple servers together can recover the original exchanged data. To achieve this aim, several approaches [74], [75] have been proposed in the literature on how to treat such a problem in a secure and efficient manner.

4) User Authentication and Access Control: In most cases, these techniques are jointly used and may restrict access to the keys employed during the notification encryption/decryption. User authentication, by means of proper information provided by the user, consists in confirming the truth about the user's identity. An identity is the representation of an entity in a particular context [76], made of a unique identifier and a set of attributes related to the user, and needs proper systems to be managed. Such systems are typically defined in the literature as Identity Manager (IdM), and have the responsibility of controlling the life-cycle of identities, verifying the truthfulness of claimed identities and exchanging identity attributes with peer systems. Over the years, several different models for representing identities and architectures to structure IdM have been proposed. None of them is an ideal solution, but each has its own advantages and disadvantages. The *password-based identity models* are the most simple and most commonly used: each user has a unique identifier in the system and a given password, to use to perform his/her authentication. Specifically, the user securely provides his/her identifier and password to IdM. If IdM finds a match between the received and the stored information, the authentication is successful and the user receives in return an access token. Such a token is an opaque string that temporally identifies a user and can be used to access the services which trust the IdM releasing it. The most well known example of password-based authentication is Kerberos [77]. Password-based authentication methods are known to be prone to vulnerabilities, depending on how "good" the used password is, e.g., how easily it can be guessed. A different kind of model is the one called *certificate-based identity management*, that makes use of digital certificates and public-key cryptography. Specifically, a digital certificate is a data structure containing the user's identifier, and his/her public key and identity attributes, verified and signed by a trusted authority, named the Certification Authority, in an unforgeable format. Certificate-based solutions typically specify the data model of certificates, the communication protocols to exchange and manage certificates and the process of issuing certificates. After a user has obtained a certificate, he/she can give it to a service provider, which can verify its authenticity by interacting with the authority that has issued

TABLE IV
MAIN ACCESS CONTROL SCHEMES: PROS AND CONS

Scheme	Pros	Cons
ACL	Simple Implementation	Scalability and Expressivity Limitations
RBAC	Grouping Rights into Categories	Coarse-grained Scheme
ABAC	Fine-grained Scheme	Issues with Heterogeneous Organizations
PBAC	Standardized Access Rules	Complicated Model

it. If the certificate is verified as being authentic, then the user is authenticated. The most well known examples of certificate-based solutions are X.509 [78] and Security Assertion Markup Language (SAML) [79]. Despite offering good levels of security, such solutions may prove to be very expensive and cumbersome. IdM can have three possible architectures [80]: (*i*) *Isolated*, where the service provider also acts as IdM; (*ii*) *Centralised*, where a single IdM is present to manage identities for a set of service providers within a given domain; and (*iii*) *Federated*, where several IdMs are used within a federated trust domain so that the identities from different domains (i.e., those issued by different IdMs) are recognized over all the domains so as to realize the Single-Sign-On (SSO). The first architecture is simple to realize, but not so scalable since a user has to manage as many identities as the providers he/she intends to interact with. The second architecture allows the management of more services with a single identity, but has problems since the centralized IdM represents a single-point-of-failure and a performance bottleneck. The last solution is more scalable and efficient, but is more complex to be realized.

Access Control disciplines which resources of the publish/subscribe service a given node is authorized to access, so as to implement authorization. During the last few years, several different access control models have been proposed, which have been summarized in Table IV. The most basic access control method is the one known as Access Control List (ACL) [81], sometimes known in the literature as Identity-Based Access Control (IBAC), consisting in a list of permissions attached to a resource of the system to be protected. A practical example of an ACL within the context of a publish/subscribe service can be to define topics as resources, to indicate the operations a client can make on the topic as permissions (namely, the creation or deletion of certain topics, and the publishing or consuming of notifications related to a given topic), and to make a list by indicating for each user which operations are allowed for each topic in the system. ACL provide a simple solution to access control, but they present several drawbacks, especially when a large number of users and permissions are needed to be managed. A more advanced method than ACL has been designed so as to overcome these limitations, where the access to a resource is based on the roles that individuals play within the organization in control of the resource. Such a solution is called Role-Based Access Control (RBAC) [82] and substitutes the resource-focused approach of ACL with the approach of grouping individuals sharing the same permissions into the same category of people fulfilling a particular role. This is more scalable since it is not necessary to indicate all the permissions of the individuals to access each resource, but only the ones related to a role, associating individuals to one, or even more, roles. Regardless of its advantages, RBAC also suffers from several limitations, such as a coarse-grained access control and an unsuitability for cross-domain accesses. To achieve a more fine-grained access control than RBAC, Attribute-Based Access Control (ABAC) [83] has been introduced. In such a model, grants to access certain resources are decided based on the attributes possessed by the requester, the context and/or the resource itself. The benefit of using ABAC is the absence of the need to know the requester in advance (as required by ACL): as long as the provided attributes match with the requested ones, access is permitted. ABAC can meet some issues in a large environment where disparate attributes and access control mechanisms exist among the organizational units. To have a more harmonized access control across the enterprise and to achieve a more uniform access control model than ABAC, Policy-Based Access Control (PBAC) [84] has been proposed, where the attribute based approach of ABAC is standardized by defining access rules in terms of policies written in a formal and precise language (such as the eXtensible Access Control Markup Language (XACML) standard [85]), which allows policy negotiation among the distinct units of an enterprise or even different federated enterprises [86]. The above mentioned models are those most studied and successfully used in several domains (as a concrete example, IHE indicates in [87] PBAC as the access model of reference for the healthcare domain, and OASIS has issued an XACML profile tailored on the healthcare domain [88]); however, access control is still an open research topic.

5) Secure Routing: It guarantees that all the peers and brokers are always interconnected by preserving routing-related information, and notifications always reach their destined subscribers. Such a technique encompasses mechanisms for intrusion tolerance so as to countermeasure threats such as the unauthorized creation/alteration/destruction of notifications or even denial of service attacks. Secure routing should not be confused with reliable routing [10], which provides the same guarantees under different assumptions: the network and nodes can manifest unintentional failures that negatively affect the delivery guarantees or even isolate certain nodes from the rest of the publish/subscribe service. The assumption of secure routing is to have non-faulty channels and nodes, but there may be some attempts by adversaries to attack the service, which should not have any consequences on the notification delivery and system connectivity. In fact, the problem is that of minimizing the effects of misbehaving, e.g., Byzantine or selfish nodes, which intentionally deviate from correct behavior so as to improve their own situation (i.e., being selfish) and/or cause problems for the system (i.e., being Byzantine). Realizing a secure routing does not only mean applying one of the known solutions for reliable routing, such as those described in [10], but a proper means is needed to deal with a Byzantine behavior of a node. As a concrete example, let us consider a well known reliable routing solution named gossip dissemination [89], also used to make a reliable event notification in [90]. Such a solution consists of a node randomly selecting a certain number of other nodes and distributing data so as to tolerate message losses due to fault manifestation. The non-deterministic selection of nodes is a weak point that Byzantine nodes can exploit if they are bent on gaming the overall system. To achieve this aim, gossip dissemination uses a verifiable pseudo-random partner

selection mechanism so as to handle Byzantine nodes and to eliminate non-determinism, as presented in [91].

6) Auditing: It consists of collecting, storing, and distributing all the information related to the received requests and consequent outcomes (jointly with the respective identity of the user that has requested the operation). Audit represents a pivotal element to achieve a high degree of security by allowing the detection of attempted security violations aimed at breaking the mechanisms used to protect the overall system and promotes an improvement of the security mechanisms to avoid future successful violations.

D. Security Terminology

To better understand the above-described mechanisms and the possible attempts at breaking them and compromising the security of a publish/subscribe service, it is worth surveying some relevant definitions of the basic security concepts and terms. Such terms are commonly used in academic and industrial works on security, and sometimes they are even confused with each other (being interchangeably adopted). In the following list we provide a set of the most known terms and their well-established definitions, while Fig. 6 depicts the dependencies among them. Here, a definition of these terms have been taken from standard glossaries, such as [92], [93]:

- A *Threat* is defined as any circumstance or event that may have a negative impact on, or even damage a system, or even one of its portions, called an asset. This can occur by successfully exploiting a given vulnerability.
- An *Asset* is understood as the system resource to be protected and preserved from possible threats.
- An *Incident* is a result of a successful threat with a violation of the system's security requirements.
- A *Requirement* concerning the security degree offered by a system specifies how the system should behave considering the above-mentioned primary and secondary attributes of the security.
- A *Vulnerability* is assumed to be a weakness of the system that can be triggered by a threat and upon which an adversary can perform an attack.
- An *Attack* consists in any kind of malicious activity conducted by an adversary that attempts to cause a threat for a given system by exploiting a certain vulnerability.
- An *Adversary* is a malicious user of a given system with the intention of causing incidents by performing attacks on the system's vulnerabilities.
- A system can be equipped with a certain set of *countermeasures* that may have two main goals:
  - — Resolving vulnerabilities so that adversaries cannot exploit them to cause threats (*Intrusion Avoidance*).
  - — Detecting and removing threats as soon as they manifest within the system and before they can cause any incidents (*Intrusion Tolerance*).

E. Security and Dependability

Security is sometimes associated with dependability. As a matter of fact, in [94] security is considered as an attribute of dependability; in [95] security is interpreted from the dependability view point, while in [96] (and mostly accepted within the current literature), security is aggregated with dependability so as to have a composite term called "Dependability and Security". Techniques to provide reliability and fault-tolerance typically also protect against deliberate attacks. In fact, the current solutions aimed at providing loss-concealing and self-correcting capabilities have the side-effect of augmenting the cost that an adversary has to pay to perform an attack. As a consequence, in the current security literature we can also find terms taken from dependability, such as the widely-known chain of fault, error, and failure (where one is the cause of the following element in the chain):

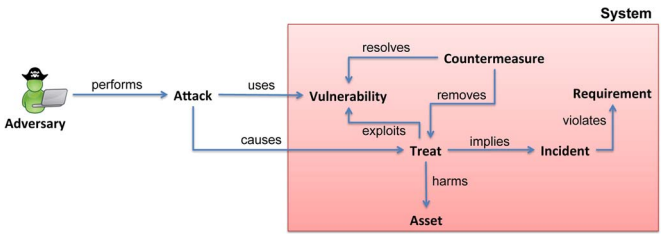

Fig. 6. Typical security terms and their dependencies.

- A *Failure* is defined as the situation when the obtained behavior deviates from that expected of a system.
- An *Error* is understood as the part of the system state that may lead to a failure.
- A *Fault* is the cause of an error and is identified as a defect in a system.

Despite the fact that the terms may seem the same, there is an intrinsic difference between dependability and security depending on whether faults happen by accident or are caused deliberately. In fact, dependability considers faults to be random and mostly independent; however, security assumes them to be intentional and malicious since they are the effects of deliberate attacks conducted by adversaries to compromise the target system [96]. This class of faults is typically named as Byzantine (after [97]), and in certain papers it is possible to find references to Byzantine Fault-Tolerance (BFT). Reference [98] describes in detail the Byzantine behaviors of publishers, subscribers, and brokers, which are included in the attacks we have pointed out in Fig. 6. The solution proposed to deal with this kind of attack is by means of the replication techniques successfully applied to make systems fault-tolerant. Specifically, the effects of Byzantine behaviors are masked by replicating the brokers in the notification service. In our opinion, supported also by [99], guaranteeing BFT allows the system to tolerate a number of its components being compromised by a security violation, so as to keep high the overall availability of the system. For this reason, we include such replication techniques within the context of secure routing, without making any further reference to BFT.

III. Secure Publish/Subscribe Services

Security for publish/subscribe systems has only recently been made the object of intense investigation by several research groups around the world. We have collected, by using Google Scholar, IEEExplore and ACM Digital Library, the

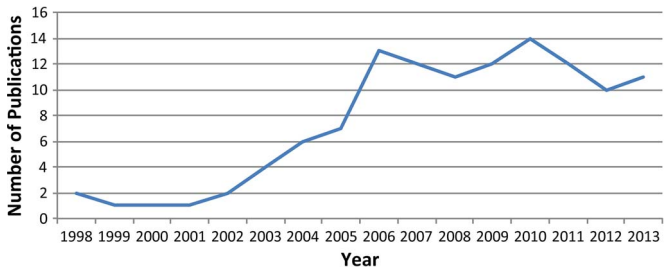

Fig. 7. Articles on secure publish/subscribe services over the years, accessed at digital libraries.

publications on this topic presented at international conferences or available in the main international journals in the last decade. When an approach has been published in several papers in different years with the proposal of refinements and improvements, we have counted it as a single paper in the median year of the distribution. The result of this search is depicted in Fig. 7, which does not pretend to be exhaustive and accurate, but helps to show an increasing trend in the literature, especially in the last eight years. This proves a great recent interest in the issue of security in publish/subscribe services due to its criticality for the success of large and important industrial projects adopting such services as their means of communication.

A. Security Requirements

Security is a well-established concept within the current literature, and with respect to publish/subscribe services it can be formulated as the ability of only authorized publishers to distribute notifications within the service and of only authorized subscribers to be able to access the published notifications of interest. Based on the taxonomy in [96] and the security requirements stated in [100], we can better define it as a combination of three properties, commonly indicated as the *CIA Triad*:

- *Confidentiality*: the published notifications are not made available to unauthorized subscribers, even if they match their interest.
- *Integrity*: any manipulations of the notification content and the overall service functionalities only happen in a standard and authorized manner.
- *Availability*: the published notifications are not denied to authorized subscribers when they match their interest.

The CIA triad represents the primary attributes for the notion of security; but such attributes are not felt to be sufficient to completely define all the properties that a generic publish/subscribe service has to provide to be considered secure. In fact, security professionals have felt that the focus on confidentiality, integrity and availability alone is not sufficient to satisfy the requirements of protecting sensitive information. For this reason, secondary attributes can be defined [96], which refine or specialize the primary ones so as to better characterize the security concept:

- *Authenticity*: The identity of users has to be validated (i.e., User Authentication), and a message has to be verified to have not been altered on its way from the publisher to a given subscriber (i.e., Message Authenticity).

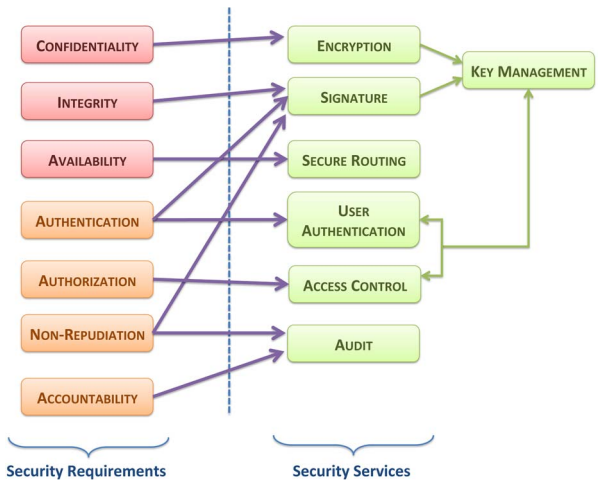

Fig. 8. Mapping between requirements and the needed services.

- *Authorization*: The rights to access certain notifications or even to use certain service functionalities are granted to given authenticated users depending on several factors, such as their role within a given organization or properly formalized security policies.
- *Accountability*: Each activity triggered by users has to be persistently traced so as to allow later forensic analysis and to map a security threat to a responsible party.
- *Non-Repudiation*: It should be possible to have evidence on who performed certain operations within the service.

Message authenticity and non-repudiation are strongly related to integrity [101]. To have message authenticity, the received message must be identical to the one originated at the publisher, which is only assured by adopting integrity. If the publish/subscribe service provides user authenticity, and the user identity is part of the notification content, then the integrity of the notification can allow the authenticity of the publisher. Guaranteeing both message authenticity and integrity is not enough to imply non-repudiation [102]. In fact, if the notification service verifies only message authenticity and integrity, the subscriber does not have any way of preventing the publisher from denying having published certain received notifications. Let us consider the case where a malicious user captures a signed and encrypted notification and publishes it multiple times. In this case, the publisher can repudiate having published the same notification multiple times. In fact, neither message authenticity nor integrity protections allow subscribers to check the uniqueness and freshness of the received notifications [103] (i.e., notifications are new and not replayed from old interaction sessions).

B. Security Services

For each of the presented security requirements it is possible to identify a proper technique to guarantee security, as stated in [100] and clearly indicated in Fig. 8:

- Encryption supports confidentiality by preventing unauthorized subscribers to read notifications; but also it does not allow malicious brokers to leak sensitive data when notifications pass through them.

- Digital signatures guarantee the authorship of a notification by containing the identity of its publisher (so as to support message authenticity), and allow the subscriber to assess if the notification has been altered, or not, along its path from the publisher (so as to provide integrity). Moreover, digital signatures are a key method for generating non-repudiation evidence [104], and have received legal recognition [105].
- User Authentication and Access Control are respectively used to satisfy the Authentication and Authorization requirements by respectively providing proof of identity and controlling which resource and functionality a given user is entitled to access.
- Secure routing guarantees the availability by guaranteeing that the notifications reach the interested subscribers despite of possible threats caused by an adversary.
- Auditing allows the provision of both non-repudiation and accountability by storing data related to access attempts to the publish/subscribe service by publishers and subscribers and to the activities of the brokers.

C. Threats and Security Means for Event Notification

As mentioned, a publish/subscribe service is typically built on top of a group communication service, such as IP Multicast in case of the TLM or Scribe [40] that realizes an infrastructureless overlay by means of a Distributed Hash Table (DHT) protocol named Pastry [106]. The adopted group communication service exposes certain vulnerabilities that have to be properly dealt with, e.g., by adopting proper techniques [107], [108] to secure IP Multicast or techniques [109] to protect Pastry in the case of Scribe. In addition, the peculiar features of publish/subscribe services are the causes of the vulnerabilities that an adversary can use to perform an attack and to violate the service. Fig. 9 shows the three peculiarities of publish/subscribe services:

1) *Open publishing*, i.e., any application can play the role of publisher. Some publish/subscribe solutions, mostly with a content-based subscription such as Padres [110], require the invocation of an *advertise()* operation to manifest to the Notification Service the intention to publish some notifications.
2) *Open subscribing and consumption*, i.e., any application can be a subscriber and consume the published notifications of interest.
3) *Notification Service mediation*, i.e., as mentioned above, publishers and subscribers are not directly connected, but they are interconnected by the Notification Service.

Let us consider a publish/subscribe service with no security means, based on such peculiarities and taxonomies available in the literature, such as [111], [112]. We have inferred the possible attacks, and the required countermeasures.

1) Attacks due to Open Publishing: The first feature implies three kinds of attacks: (*i*) *Masquerading*, (*ii*) *Flooding*, and (*iii*) *Trashing*. In the first attack, a malicious user pretends to be another user so as to arbitrarily behave on behalf of the masqueraded user and to alter the view that subscribers have of the state belonging to the masqueraded user or to inject arbitrary

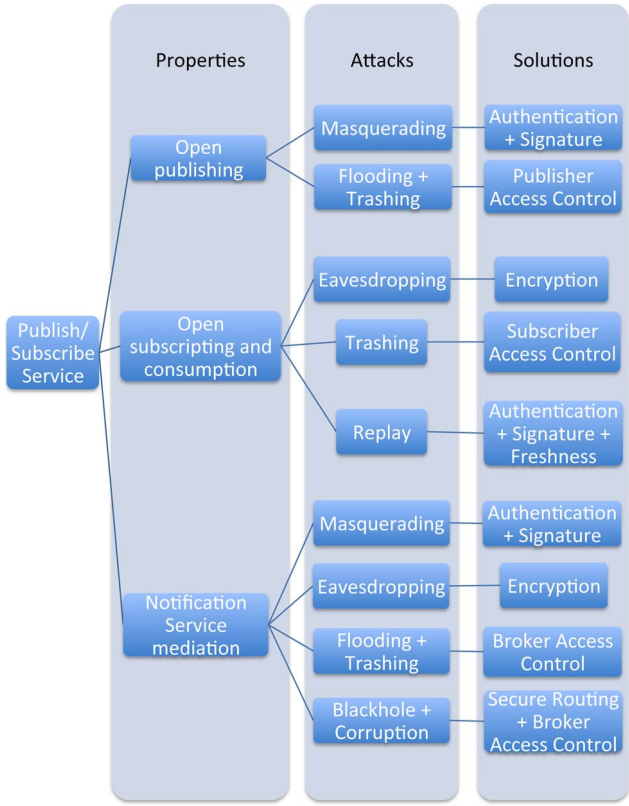

Fig. 9. Vulnerabilities and attacks derived from the peculiarities of publish/subscribe services.

messages on behalf of the masqueraded user. The solution to this kind of attack is to encompass user authentication and signature within the publish/subscribe service. In the second attack, a malicious publisher can generate a massive amount of notifications with the intention of saturating the resource within the Notification Service so affecting its availability. The third attack consists in consuming resources in the Notification Service by rapid and frequent changes of the advertisement state [111]. Both these two attacks can be addressed by introducing proper Publisher Access Control within the service.

2) Attacks due to Open Subscribing: The second feature causes three kinds of attacks: (*i*) *Eavesdropping*, (*ii*) *Trashing*, and (*iii*) *Replay*. The first consists in reading notifications directed to certain subscribers, without their consent, so as to steal the contained information. Such an attach can be resolved by encrypting the exchanged notifications. The second is similar to the ones described for publishers, with the only difference being that the attack is conducted by rapid and frequent changes of the subscription state of the Notification Service. The third attack is carried out by a malicious subscriber receiving certain notifications, which are republished with, or even without, a certain delay. Such republished notifications can affect the view that subscribers have of the state belonging to a given publisher, as described for the Masquerading. Such an attack is countered with the same solution as the Masquerading attack due to the similarities in their effects; in addition, a proper freshness scheme should be used in combination with digital signatures [56].

3) Attacks due to Notification Service Mediation: The third feature makes the service vulnerable to the following attacks:

(*i*) *Masquerading*, (*ii*) *Eavesdropping*, (*iii*) *Flooding*, (*iv*) *Trashing*, (*v*) *Blackhole*, and (*vi*) *Corruption*. A broker can perform masquerading both by faking the identity of a publisher, as above described, but also by falsifying the identity of a valid broker so as to cause a misrouting of the notifications. In both cases, masquerading is countered by user authentication, plus signatures to prevent a subscriber considering valid the notifications published by the adversary on behalf of the masqueraded user. The described eavesdropping, flooding and trashing can be easily realized by a malicious broker, and resolved with the presented solutions applied at the broker level. In addition, a malicious broker can apply two kind of attacks to alter the regular flow of notifications within the Notification Service by arbitrarily dropping invalid notifications (i.e., a Blackhole attack) or corrupting their content (i.e., a Corruption attack). We can introduce countermeasures against these two attacks beforehand by using Broker Access Control so as to avoid the possibility of a malicious broker joining the Notification Service and affecting the correct flow of notifications. However, such a countermeasure is not in itself effective to counter Blackhole and Corruption attacks and mask their effects when a malicious broker succeeds in joining the system by eluding the introduced access controls. In fact, these attacks can be addressed by using secure routing, which allows the recovery of dropped or corrupted notifications.

4) Unicast Solutions vs Group-Aware Solutions: The current literature on secure messaging is full of proposals on how to protect unicast communications, which has led to several standardized protocols, such as SSL or TLS. Such existing protocols have been proved to be effective, and may be adopted also to protect event notification within publish/subscribe services, guaranteeing the requirements we have stated in Section III-A by means of the above-mentioned methods. Despite being used by early works on secure publish/subscribe services, such a solution is ineffective for several reasons. Let us consider the case of realizing the encryption of exchanged notifications within a service using an overlay of brokers. SSL or TLS can be adopted to perform the so-called link-by-link encryption: each vulnerable communication link (i.e., from publishers/subscribers to their broker or among brokers within the notification service) is equipped at both ends with a proper protocol for encryption such as SSL or DHT. Such an approach does not satisfy the security requirements since all notifications routed through the brokers are exposed, as we cannot assume that they are trustworthy. Moreover, the existing, well-tested, transport-level unicast secure communication protocols presuppose some relationship between the communicating parties; however, in publish/subscribe services, the interactions between the publishers and subscribers are mediated by brokers, making the communication decoupled without any preexisting relationship. The opposing approach is to apply an end-to-end encryption: each publisher protects the content of its notification with a proper mechanism for encryption, and the subscribers/brokers may obtain the original data by performing decryptions. Such a solution does not need to assume that all intermediate brokers are trusted, so as to allow only authorized subscribers to have access to the received notification. The need to have an end-to-end approach and the demand for novel solutions are also motivated by performance and scalability limitations that the existing unicast secure communication protocols exhibit when applied within the context of publish/subscribe services. To have empirical evidence of this, we have compared these two solutions by implementing them within the context of a publish/subscribe service compliant to the WSN standard. Our testing application has a deployment depicted in Fig. 10(a) where there is a publisher generating 50 notifications with a size of 16 bytes and passing them to a first broker, which passes them to another broker connected to a subscriber. In a first experiment, the links between the four applications are insecure and no encryption is applied. In a second experiment, SSL is used to secure the links between the four applications. Last, in a third experiment, the links are insecure, the publisher encrypts the content of its notifications by using a symmetric encryption scheme, i.e., Advanced Encryption Standard (AES), with a key size of 192 bytes, and the subscriber decrypts them. We have compared the performance of the last two experiments with the reference case given by the first experiment, and the obtained results are depicted in Fig. 10(b). From such a figure we can notice that the use of link-by-link encryption implies a higher performance worsening than the end-to-end encryption both in the average value and in the standard deviation. It is intuitive that such a situation is further worsened if we increase the number of brokers and/or subscribers.

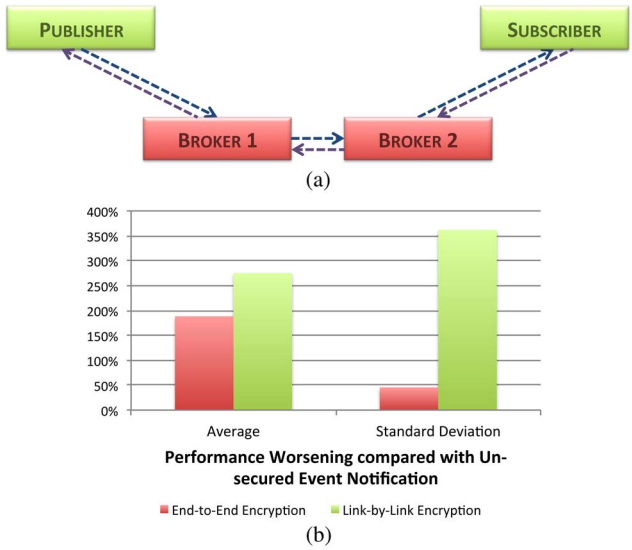

Fig. 10. Performance comparison of link-by-link and end-to-end encryption in event notification.

IV. Academic Solutions

The current academic literature on secure publish/subscribe services encompasses more than one hundred papers published from 1998 to 2014 in the main international conferences and journals. Among these, we have investigated the solutions they propose to deal with the main attacks shown in Fig. 9 by realizing the following security services: Authentication and Authorization, Secure Notification Dissemination, Secure Routing, and Auditing. Each of the following subsections presents the possible solutions available in the current literature for each of

TABLE V
MAIN ACADEMIC SOLUTIONS FOR A SECURE PUBLISH/SUBSCRIBE SERVICE

Reference	Year	Implemented?	Adopted Publish/subscribe service	Subscription Kind	Multicast Type	Infrastructure Scheme
[113]	2001	No	A generic Publish/Subscribe System	Content-based	ALM	No Details
[114]	2002	No	A generic wide-area publish/subscribe system	Content-based	ALM	Mesh Infrastructure-based Overlay
[115]	2003	Yes	The Narada Event Brokering System [116]	Topic- and Content-based	ALM	Mesh Infrastructure-based Overlay
[117]	2006					
[118]	2003	Yes	The Hermes publish/subscribe middleware [119]	Topic- and Content-based	ALM	Mesh Infrastructure-based Overlay
[120]	2006					
[121]	2007					
[122]	2011					
[123]	2004	Yes	The Rebeca publish/subscribe middleware [124]	Content-based	ALM	Mesh Infrastructure-based Overlay
[125]	2005	No	A generic Publish/Subscribe System	Content-based	ALM	No Details
[126]	2006	Yes	The Siena Publish/subscribe System [39]	Content-based	ALM	Mesh Infrastructure-based Overlay
[127]	2006	Yes	The Gryphon system [128]	Content-based	ALM	Mesh Infrastructure-based Overlay
[129]	2006	No	A generic publish/subscribe system	Content-based	ALM	No Details
[130]	2007	No	A generic publish/subscribe network	Content-based	ALM	No Details
[131]	2007	Yes	The Siena Publish/subscribe System [39]	Content-based	ALM	Mesh Infrastructure-based Overlay
[132]	2011					
[133]	2008	No	A generic publish/subscribe system	Content-based	ALM	No Details
[134]	2009	No	A generic publish/subscribe system	Content-based	ALM	No Details
[135]	2010	No	A generic publish/subscribe system	Channel- and Content-based	ALM	No Details
[136]	2010	No	A generic publish/subscribe system	Content-based	ALM	No Details
[137]	2010	Yes	The Siena Publish/subscribe System [39]	Content-based	ALM	Mesh Infrastructure-based Overlay
[138]	2012	Yes	The Padres publish/subscribe platform [110]	Content-based	ALM	Mesh Infrastructure-based Overlay
[139]	2012	No	A generic publish/subscribe system	Topic-based	ALM	No Details
[140]	2012	Yes	The Siena Publish/subscribe System [39]	Content-based	ALM	Mesh Infrastructure-based Overlay
[141]	2013					
[142]	2013	Yes	A generic publish/subscribe service	Topic-based	ALM	Mesh Infrastructure-based Overlay
[143]	2014	No	A generic publish/subscribe service	Content-based	ALM	Tree Infrastructure-less Overlay

these services. A list of all the reviewed solutions is available in Table V. All of them adopt an ALM multicast scheme, since this makes them suitable for Internet-wide publish/subscribe services where security is a concern.

A. Authentication and Authorization

Authentication and authorization are typically jointly used to realize access control, which is among the fundamental properties of a secure publish/subscribe service. For this reason, from the first papers, access control has been investigated and several possible solutions have been proposed. Table VI summarizes such solutions, and highlights their peculiarities in terms of which entities are controlled (as above mentioned, access control is needed for publishers, subscribers, and brokers), and which approaches are adopted for authentication and authorization. We have grouped such solutions with respect to the adopted access models summarized in Table IV.

1) ACL-Based Solutions: The first solution to adopt the simple ACL model is [115], which describes a simple authorization method for the Narada event brokering system [116], an open-source messaging service. Access control is performed for publishers and subscribers, without providing details on how it can be authenticated. A further evolution of this work, described in [117], indicates that clients are authenticated by means of X.509 certificates issued by a Certificate Authority within the system. Also, [127] describes an ACL authorization implemented in the Gryphon system [128], where a set of rules is associated to each authenticated client. Such rules consist of three values: (*i*) the identity of the authenticated client, the type of access (namely the authorized operations that the authenticated user is able to perform), and a content filter. In fact, the authorization rules are applied only to those events that verify a content-based filter. In addition, clients can be grouped and certain rules can be defined even for client groups.

TABLE VI
MAIN SOLUTIONS FOR AUTHENTICATING AND AUTHORIZING ENTITIES IN PUBLISH/SUBSCRIBE SERVICES

Solution	Year	Publisher	Subscriber	Broker	Authentication Approach	Authorization Model
[114]	2002	Yes	Yes	No	Credentials	PBAC
[115]	2003	Yes	Yes	No	No Details	ACL
[118]	2003	Yes	Yes	Yes	X.509 Certificates	RBAC
[123]	2004	Yes	Yes	?	Attribute Certificate	Scoping and ABAC
[117]	2006	Yes	Yes	No	X.509 certificates	ACL
[127]	2006	Yes	Yes	?	No Details	ACL
[129]	2006	No	Yes	Yes	Credentials	No Details
[121]	2007	Yes	Yes	Yes	Authorization Certificate	RBAC
[132]	2011	Yes	Yes	No	No Details	No Details
[122]	2011	Yes	Yes	Yes	No Details	Parameterized RBAC
[138]	2012	No	Yes	No	No Details	PBAC
[140]	2012	No	Yes	No	Pseudonym, Credentials and Access Token	No Details
[141]	2013					
[142]	2013	No	Yes	No	Attribute Certificates	ABAC

Therefore, the set of rules for a client is the union of the rules for the group which the client belongs to and the rules specific for the client. This solution requires the authentication of users; however, [127] does not go into detail about how authentication is performed. Moreover, the authors recognize that not all brokers can be trusted, and brokers that trust each other should be grouped in a given domain. However, the paper does not contain any details on how communications among different domains should be conducted and how to authenticate and authorize brokers in different domains that want to exchange notifications. The authors only suggest considering each domain as a special client and assigning certain rules to it.

2) RBAC-Based Solutions: The first solution adopting RBAC was that described in [118], where RBAC is applied to Hermes [119], a scalable event-based middleware developed at the University of Cambridge. In this solution, authentication is performed by means of credentials that publishers and subscribers send to the brokers along with the notifications to be

published or subscriptions to be registered. The authorization to perform a given operation (e.g., publishing a notification) depends on which role is activated by the client; however, it is possible to have more complex cases where authorization decisions can depend on the event content. Unlike the previous approach described in [118], it is stated that in a large-scale publish/subscribe service it is not possible to assume a network of trusted brokers, while it is possible to have certain brokers not authorized to manage certain types of events. Using certificate chains is proposed so as to obtain a web-of-trust creating a sub-graph of brokers trusted to manage a certain event type. Specifically, each broker has a X.509 certificate, which can be signed by the owner of the event type, or other brokers that are already trusted for the considered event type. It is assumed that all the publishers and subscribers hold the trusted root certificate of the owner of the considered event type, so as to infer if their local broker is allowed to manage the given event type. Building certificate chains in a large-scale setting can be troublesome, and assuming a unique owner for a given event type can lack scalability. This work is further extended by [121] with the introduction of authorization certificates: a quintuple indicating which capabilities the holder is able to perform in the publish/subscribe service. Such a certificate can allow the holder to delegate to another entity the capabilities granted by this certificate. A further improvement of this approach is presented in [122], where RBAC is parameterized by considering also the context.

3) ABAC-Based Solutions: A different approach, based on ABAC, is described in [123], which presents the security solution applied to a publish/subscribe service developed at the Technische Universität Darmstadt, namely Rebeca [124]. In this paper, the scope of the proposed solution is to keep the communications within the group separate (namely all the subscribers interested to a given topic, and the publishers relative to this topic) from outsiders, and use credentials to establish mutual trust among the group members. The driving idea of this solution is to exploit scoping [143]: the visibility of the notifications is limited to the scopes for which they have been published. Scopes are intended as a means to isolate intra-scope traffic from the rest of the world and provide security. Specifically, client access control is realized by means of Attribute Certificates [144], which hold client identity and a set of attributes, specifying if the client is authorized to subscribe to or publish certain events. Such a certificate can be issued by the administrator of the service or by another trusted attribute authority. The certificate is verified by the border broker, by checking its signature; if it matches the attributes specified by the entity that signed the certificate, then the requested operation can be performed. The infrastructure managing the scopes would have to be trusted, and in [123] it is presented a distributed approach to build an overlay for each scope within the service. However, how a contacted trusted broker accepts or not the joining request coming from another broker is not described, since it is considered application- and scope-specific. Furthermore, it is not possible that all the members of a scope can be reached by trusted brokers; therefore, a tunneling method is presented so as to transport notifications through untrusted brokers by encrypting the notification content. Also, [141] adopts ABAC by extending the approach presented in [123], where each message has associated attributes on the identity of the publisher and has such attributes checked so as not to route them where they are not necessary or authorized (so as to integrate access control within the routing decisions of the brokers).

4) PBAC-Based Solutions: The very first work on access control in a publish/subscribe service proposed the use of PBAC. In fact, in [114], the publishers and subscribers are authenticated by means of credentials, consisting in digitally signed documents or certificates that contain the user attributes and claims. Such credentials are associated to each request made at the notification service by the publishers and subscribers. The contacted broker holds a policy list so as to verify if the requester is allowed to invoke the requested operation. Such policies are expressed in a similar manner to subscriptions, i.e., with content-based predicates. The limitation of this approach is its assumption of a network of trusted brokers. Another work adopting PBAC is presented in [138], which aims at enforcing access control with encryption, by presenting a solution for the publishers to express which subscribers can access their published events through the use of Ciphertext-Policy ABE (CP-ABE) encryption [145].

5) Generic Solutions: The remaining solutions listed in Table VI do not provide precise details on the adopted authorization approach. [129] proposes an approach where brokers register at publishers and subscribers register at their nearby brokers, taken from an authorized list of brokers distributed by the publisher. At each registration, the credentials of the requesters are verified so as to perform authentication. Such a registration also performs authorization, and no details are provided on how it is performed. The framework named as EventGuard, presented in [132], represents an advanced solution to support security in publish/subscribe services. It consists in a set of security guards to be plugged into a given content-based publish/subscribe service. Among several interesting features, it provides a solution for the authentication and authorization of clients by including an access control engine that determines which subscriptions or which advertisements are authorized. Access control is enforced by passing the right set of encryption and decryption keys to the authorized subscribers and publishers, respectively. However, there are no details on how the access control engine works and which approach is adopted to authenticate clients and authorize subscriptions and advertisements. Last, [139] illustrates a solution for subscriber access control where authentication is performed twice by the subscribers: (*i*) the subscriber registers at the publisher by giving its unique id, which can be a pseudonym, and other credentials, and obtains an access token and symmetric encryption key; (*ii*) the subscriber registers subscriptions at its local broker by using the access token obtained from the publisher.

6) Solutions for Authentication: Despite the fact that great efforts have been made to develop access control models and authorization solutions, very little attention has been devoted to authentication. As shown in Table VI, around half of the surveyed solutions on access control do not provide any details on how authentication is provided. Among the others, [114], [129] mention using credentials for authentication, but with

TABLE VII
MAIN SOLUTIONS FOR APPLYING NOTIFICATION ENCRYPTION IN PUBLISH/SUBSCRIBE SERVICES

Solution	Year	On Notifications	On Subscriptions	Scalable Key Management	Applied Encryption Scheme	Matching Encrypted Data
[113]	2001	Completely	None	No, periodic group rekeying	Symmetric cryptographic scheme	None
[115]	2003	Completely	None	No, periodic group rekeying	Symmetric cryptographic scheme	None
[123]	2004	Completely	None	No	TLS	No
[125]	2005	Only certain values	None	Yes, with performance costs	Asymmetric proxy re-encryption	None
[126]	2006	Completely	Completely	No, using group key	Symmetric cryptographic scheme	Yes
[129]	2006	Completely	None	Yes, using key trees	Symmetric cryptographic scheme	None
[131] 2007 [132] 2011		Completely	Completely	No, with hierarchical key derivation and heavy rekeying	Asymmetric cryptographic scheme	Limited Expressivity
[130]	2007	Completely	None	No, centralized key server	Asymmetric key traitor tracing scheme	None
[134]	2009	Completely	Completely	No, several keys to manage	Multiple layer commutative encryption	Limited Expressivity
[135]	2010	Completely	Completely	No, per-subscriber keys	Symmetric cryptographic scheme	Limited Expressivity
[136]	2010	Only certain values	None	Yes	ABE	None
[137]	2010	Completely	Completely	Not scalable	SDE	Yes
[138]	2012	Completely	Completely	Yes, with performance costs	ABE and multi-user SDE	Yes
[140] 2012 [141] 2013		Attribute names not encrypted	Attribute names not encrypted	Yes, with performance costs	Pailler homomorphic cryptosystem and symmetric group keys	Limited Expressivity
[143]	2014	Completely	Completely	Yes	PBC	Yes

no further details on how such credentials are structured or issued. The certificates are used by the remaining solutions; however, password-based authentication is not adopted in the current academic works on authentication in publish/subscribe services. The first work to use certificates is [117], where X.509 certificates are issued by a central certification authority within the system. References [118] and [121] adopt the Open Architecture for Secure Interworking Services (OASIS) framework [146], where certificates are modeled according to the X.509 standard, but are used as Authorization Certificates, and the acquired capabilities and permissions can be delegated. These two works are different from that in [117]: the certificates are issued by a certification authority and used as a proof of the identity of its holder. The certificates in [118], [121], known as public key certificates, are issued by an attribute authority, and used to characterize or entitle its holder by indicating its allowed capabilities. The advantage is that these certificates convey authorization information among distributed entities. Moreover, the attribute authority is not centralized in the system, and it is possible to realize delegation or substitution over multiple systems thanks to certificate chains. Public key certificates support neither delegation nor substitution. Also in [139] we have attribute certificates, which have been made anonymous so as to avoid the leakage of personal information of the owner. However, no details are provided on the IdM management. Instead, [141] provides precise details on how the IdM management is performed. Specifically, it makes use of attribute certificates, a federated IdM framework called Gismo [147] and established trust relations among identity providers so as to realize SSO. Moreover, it employs trusted bindings, introduced in [148], to reduce the number of authentications to be carried out: if an application A (publisher or subscriber) trusts a broker B, which trusts in turn the broker C, then A will also trust C.

B. Secure Notification Dissemination

The techniques for secure notification dissemination are those that aim at guaranteeing confidentiality, integrity, notification authenticity and non-repudiation. As shown in Fig. 8, such techniques consist in encrypting and signing the notification content, so as to avoid the eavesdropping of the sensitive information contained in the notifications, their malicious manipulation, or the possibility of replaying them.

The adoption of a cryptographic scheme in publish/subscribe services presents some requirements [138]:

- Notification Confidentiality—The content of the exchanged notifications has to be protected by means of encryption and only authorized subscribers should be able to decrypt it and have access to the original sensitive information carried by the notifications.
- Subscription Confidentiality—Malicious adversaries could infer sensitive information and/or subscriber's intentions by eavesdropping the subscriptions exchanged by the subscribers with the notification service, and therefore the subscription predicates should be encrypted as well.
- Scalable Key Management—The publishers and subscribers should be kept decoupled, so that they are not allowed to share keys so as not to weaken their decoupling degree; to avoid this, a proper mediating approach should be adopted.
- Encrypted Matching—in the case of content-based publish/subscribe services, the brokers have to access notification content so as to take any routing decision by matching such a content to the content-based predicates submitted by the subscribers; however, this could, when brokers cannot be trusted, lead to information leaks. Therefore, it is necessary for brokers to take content-based routing decisions without revealing such confidential information so as to avoid eavesdropping cases.

1) Link-by-Link Encryption: Some solutions, such as [123], establish long-term pairwise secure channels among end-points within the notification overlay by using SSL or TLS; however, this is not an efficient solution. In fact, we have provided empirical evidence on the inferiority of a link-by-link approach to an end-to-end one in Fig. 10. Therefore, novel approaches have been proposed during the last few years, an overview of which is shown in Table VII.

2) Symmetric Cryptography With Per-Subscriber Keys: [135] proposes a symmetric encryption scheme to protect notifications and subscriptions by allowing publishers and subscribers to share a set of secret keys. Specifically, all the

published events are encrypted by using these keys, and all the submitted subscriptions are also encrypted. In addition, a random number r is generated by the publisher and added to all the numeric values so as to hide the real value. Such a random value is distributed to all the subscribers in a secure manner (i.e., by encrypting it with the subscriber key). The brokers are not aware of the adopted keys and random number, and cannot access the original information and perform matching on the encrypted data with encrypted subscriptions sharing the same key. Such a solution is not scalable since the publisher makes several copies of a given event, each encrypted with the subscriber key. Moreover, the publisher has to establish a direct connection with the subscriber so as to obtain its key. Such an approach only supports equality and ordering relations to perform content-based matching, and subscriptions can be composed of only one relation.

3) Symmetric Cryptography With Group Keys: The first works in the literature to handle encryption within a publish/subscribe service are those described in [113], [115], which have drawn from the literature on secure group communication services [149]. Such solutions aim at reducing the number of encryptions to be performed than is the case in the previous cryptographic schemes. The solution is to have group keys, i.e., all the communications within the group are encrypted with a given key, used also for the decryption.

Group keys are simple to use when the publish/subscribe service is able to clearly define a group, such as the case of topic- or type-based subscriptions. As a concrete example, [115] uses group keys to protect notifications exchanged with the Narada Brokering system [116], where the group is identified by the topic associated to each notification. However, it is less intuitive to adopt such solutions with content-based subscriptions, where groups are dynamically defined based on the current content of notifications. Reference [113] proposed a solution to this issue by assigning proper group keys to a certain subset of subscribers, and by using several cashing approaches. Such a solution is based on the assumption that several events should be delivered to the same subset of subscribers. Such subsets are considered as groups with the same key. Such an approach can be used with any symmetric cryptographic scheme, and it is only applied to notifications. Finally, both in [113], [115], content-based subscriptions are evaluated prior to the encryption of the notification content, so there is no matching on encrypted notifications. Reference [126] describes a solution using group keys implemented in Siena [39], where both notifications and subscriptions are encrypted by using a symmetric scheme with the definition of a group key. The difference from the previous solutions is that it provides matching on encrypted data by means of two distinct techniques: equality tests based on the scheme in [52] and the keyword matching from [150].

One of the main properties a group key management system has to enforce within a notification service is secrecy: only members of the group are authorized to have access to the notifications exchanged within the group. Secrecy can be characterized by means of two aspects: the backward secrecy and the forward one. In the first case, a member should not be able to obtain the notifications exchanged before joining a group. In the second case, a member should not be able to obtain the notifications exchanged after leaving the group. These two secrecy aspects are accomplished by means of proper rekeying mechanisms, where the adopted keys are periodically changed and notified to the group members. Such rekeying mechanisms are affected by many drawbacks. Traditional rekeying schemes need N encryptions and accordingly N transmissions, where N is the number of members of a given group [149], causing a very high communication overhead and rekeying delay. Moreover, traditional rekeying schemes are typically individual, i.e., triggered when a membership change occurs within the group (e.g., when a new member joins the group or an existing member leaves). This incurs in the 1-affect-n problem: a single change has effects on all the members of the group, with severe effects on the scalability of the overall communication infrastructure. Last, in the case of individual rekeying schemes, we can also have out-of-sync problems compromising the offered secrecy level, e.g., a member that left the group is still capable of decrypting a newly published notification since the publisher has not received the new key on time before publishing a new notification. In the final analysis, despite the simplicity of this approach, the scalable key management is not applied with classic group key schemes due to the highly computational overhead of updating the group keys and the out-of-sync problem.

The logic tree-based schemes proposed in [151] can reduce the management costs of group keys (making them increasingly logarithmic according to the greater group size); however, the communication overhead and rekeying delay still remain high in the case of a large scale network. In addition, a central key manager has to monitor the status of all group members, and keep a fully-connected topology of all the trusted members, implying a high management overhead. Even if a distributed solution is applied, the key management does not result more scalable due to the fact that the storage cost for each node and the management overhead to monitor other trusted members augment linearly for greater group size. A concrete application of such a solution is in [129], which describes a decentralized group key management protocol, such as Iolus [152]. Specifically, the overall network of brokers, subscribers, and publishers forms a hierarchy of sub-groups: a certain set of subscribers directly connected to a broker makes a sub-group, and all the brokers and the connected publishers build up another kind of sub-group. There are three kinds of key pairs that certain members of the network hold so as to perform decryption. A common key pair is used to encrypt and decrypt notification content. Such a key pair is owned by the publisher and the relative private key is made available to the brokers and subscribers. In addition, each sub-group with a broker and certain subscribers has its own key pair, generated by the broker and used to secure the communications within the sub-group. The private key of such a pair is distributed to the subscribers of the sub-group. Finally, each node of the tree, namely the brokers and subscribers, has its own key pair, used to protect the distribution of the common key. This approach is more scalable since changes and faults within a sub-group do not propagate to the other ones. However, it has the problem that the data passing from one sub-group to another have to be translated

from a group key to another one, causing a performance penalty. Such a transformation is managed by the sub-group broker, which can be a bottleneck. Last, it does not deal with matching encrypted data.

4) Asymmetric Cryptographic Schemes: A large number of the solutions in Table VII use asymmetric cryptographic schemes, despite their being less efficient than the symmetric ones. Such inefficiency is a price worth paying, since the system is repaid with the peculiar ability of public-key cryptography to jointly provide user authentication and secrecy (when the notification is encrypted first with the publisher private key and then with the subscriber public key), while a symmetric scheme offers only secrecy. Moreover, sharing a common key, as occurs in symmetric cryptographic schemes, contradicts one of the key properties of publish/subscribe services: the decoupling of publishers and subscribers. The first solution to avoid using group keys and achieve a more scalable key management is proposed by [125], where events are structured as XML documents, with some attribute values, considered sensitive and not used for the content-based routing decisions, kept confidential by using an asymmetric cryptographic scheme. In addition, along the path from publishers to subscribers the brokers apply the proxy re-encryption scheme of Jakobsson [73]. In this manner, the subscriber receives a notification it can decrypt with its own private key. Such a solution is only limited to notifications (and only to certain attribute values), without protecting the confidentiality of the subscriptions. The publishers and subscribers do not have to share anything so as to achieve a scalable key management. However, the proxy re-encryption scheme implies some non-negligible performance worsening due to the necessary decoding and re-encryption operations performed by PSAS for each subscriber. Last, as above mentioned, the proxy re-encryption scheme of Jakobsson is not collusion-safe.

The approach in [131] (extended later in [132]) completely encrypts notifications and subscriptions by an asymmetric scheme consisting of encryption and authorization keys. As stated in their name, the first key is used by the publishers to encrypt the notification content, while the second one is used by authorized subscribers to access the original content of the received events. These two keys are built by using the hierarchical key derivation algorithms [153]. The keys used to protect the notifications exchanged through the notification service are structured as a hierarchy, and the users can express numeric attribute-based subscriptions (i.e., range conditions on a given numeric attribute of the exchanged notifications). Each authorization key is associated to a subscription, and an encryption key for a given event is built from the authorization key: it is computationally easy to derive the encryption key if the event satisfies the subscription associated to the authorization key; on the contrary this is computationally hard if the event does not satisfy it. These keys are properly mapped to a common key space using a numeric attribute key tree. This is, however, not a scalable key management approach, since when a user invokes the unsubscribe command, then it is necessary to perform a rekeying so as to have new keys and prevent a subscriber that has left the system being able to read future notifications. Moreover, the expressivity of the applied content-based subscriptions is limited to range conditions on a single numeric attribute.

Another asymmetric solution is the one in [130], which adopts the public-key traitor tracing scheme described in [154], where each subscriber has a private key to access the content of received notifications, and a traitor tracing algorithm is able (*i*) to detect any traitors, (*ii*) to revoke their private keys, and (*iii*) to avoid updating any private key of the remaining subscribers. Such a solution encrypts only notifications, assumes a centralized key server, and ignores the issue of matching on encrypted data. This last problem is not neglected by [137], which proposes a different asymmetric approach based on Asymmetric Scalar-product Preserving Encryption (ASPE), which allows brokers to verify subscriptions without accessing the original information. Such a solution supports several kinds of subscription conditions, which are representative of a generic content-based publish/subscribe service. The publishers and subscribers share an invertible matrix for the transformation of notifications and subscriptions.

References [136] and [138] propose a scheme based on ABE; but only [138] deals with encrypting subscriptions and processing encrypted data by applying multi-user SDE. Such a scheme allows multiple users to encrypt data and perform queries without sharing keys, and allows brokers to perform any typical content-based predicates without discovering any sensitive information. Such a solution is built on top of the above mentioned proxy re-encryption schemes. The solution in [138] protects both notification and subscription confidentiality, while [136] only encrypts sensitive information not used to evaluate subscriptions. In both cases, confidentiality is achieved without requiring publishers and subscribers to share any secret keys.

The last asymmetric solution is that proposed in [142] where the chosen cryptographic scheme is PBC. The solution in [49] offers subscription confidentiality, but only in a weak form. Specifically, the underlying publish/subscribe service is based on a Tree Infrastructure-less overlay, so the knowledge of the subscription of the parent and children of a node in the tree is crucial to realize the maintenance of the overlay topology. Additionally, such a solution allows the realization of matching on encrypted data, thanks to the Public-Key Encryption with the keyword Search described in [155].

5) Generic Cryptographic Schemes: Some models have been presented, without indicating if the cryptographic scheme is symmetric or not. A concrete example is in [134], presenting a novel approach for the matching and content-based routing of encrypted notifications. Such an approach based on the special variant of proxy re-encryption is named as the commutative multiple layer encryption scheme [156]. The driving idea of this approach is to allow brokers to perform transformations without accessing the original sensitive data contained in the notifications as cyphertext. This is possible by adding and removing several encryption layers, under the consideration that the encryption mechanism is commutative, i.e., given two distinct keys, namely k_1 and k_2, and any data of interest, namely d, encrypting d first with k_1 and then with k_2 is equivalent to encrypting d first with k_2 and then with k_1. The solution in [134] works by making the subscriber encrypt its subscription

TABLE VIII
DIGITAL SIGNATURES IN PUBLISH/SUBSCRIBE SERVICES

Solution	Year	Freshness	Signature Method	Note
[125]	2005	No	XML digital signatures	Two signatures on notifications: one computed on the encrypted data and one obtained from the plaintext.
[130]	2007	Yes, with nonce	No details	Signatures not only to notifications, but also to all other requests.
[120]	2007	Yes with timestamps	No details	Signature also applied to event types.
[133]	2008	No	MSS	Signature of only notifications.
[136]	2010	No	Sanitizable signatures	Signature verified even if a part of the message is not known.
[132]	2011	Yes, with timestamps	ElGamal signature	Subscriptions and advertisements are signed by a third trusted service, while publishers signs their notifications.
[142]	2013	No	No details	Signatures on notifications.

with $r \geq 2$ encryption layers, and the publisher encrypt its notifications with the same number of layers. The brokers placed en-route between the publishers and subscribers remove one layer and add another one so as to have the notification content always protected by at least $r-1$ layers. The number of layers indicates also how the main neighboring nodes in the service topology need to share the adopted symmetric keys. This solution implies the management of a large number of keys, making it not very scalable. Finally, the content-based routing decisions are taken by equality matching the encrypted values in the notification with encrypted subscriptions.

Reference [139] presents an approach based on both symmetric and asymmetric scheme. Specifically, a notification is characterized by a payload encrypted with a symmetric key and a set of blinded values encrypted with an asymmetric key. Such values are used to perform content-based routing, and a Pailler homomorphic cryptosystem is used so as to perform privacy preserving matching over encrypted data. This solution offers very simple content-based predicates for the subscriptions, keeps the attribute names clear (only their values are blinded), and ensures that the subscribers need to register at publishers and share the symmetric keys with them. In addition, the solution presented in [139] has the drawback that each subscriber has to submit a new subscription via a secure channel to a TTP that encrypts the subscription. Such a process, named as blinding, needs to be redone every time subscriptions have to be updated, i.e., whenever the context of the subscribers changes. This flaw has been resolved in [140] by allowing authorized subscribers to obtain some public security parameters at the time of registration to derive the adopted keys, and to contact a TTP only when the public security parameters are updated. This solution propose to have two sets of encrypted values for each notification, implying a considerable performance degradation.

6) Solutions for Digital Signatures: Even if the digital signature is important to detect masquerading or replay attacks, very few academic solutions for secure publish/subscribe services have applied it, the main ones being listed in Table VIII. The issue is that the signature is typically used in publish/subscribe services when the message content is partially or completely encrypted. Therefore, it may be necessary not to let the recipient access the plaintext of the received message, which may carry sensitive information, to verify the correctness of the signature.

A solution proposed in [125] is to attach to the exchanged messages two signatures: one computed on the encrypted portion of the messages so that untrusted destinations can verify the signature without leaking sensitive information; and one obtained from the plaintext of the messages so that authorized destinations can test the integrity of the received messages. Such signatures are expressed according to XML digital signatures [157], and are applied to both notifications and to any user requests. The approach in [141] also uses signatures on notifications, but it does not provide any details about the applied signing scheme. The solution proposed in [130] applies the signatures not only to notifications, but also to all other requests that a notification service can receive (i.e., *subscribe()*, *unsubscribe()*, *advertise()*, *unadvertise()*). The use of two signatures, proposed by [125], seems wasteful and could cause performance worsening. To resolve this, [136] proposes the use of sanitizable signatures. EventGuard described in [132] also adopts signatures to protect both notifications and all other requests, expressed as a tuple $\langle r, s \rangle$, where r represents a unique identifier of a certain notification and is used to detect possible duplicates and to avoid replay attacks, and s contains the signature. The adopted solution is a probabilistic signature, specifically the ElGamal signature algorithm. In [132], signatures for subscriptions and advertisements are created by a third trusted service, while publishers create signatures for their produced notifications. Reference [133] proposes a signature scheme based on the MSS.

The work presented in [120] applies signature schemes also to event types when a new type is defined within the publish/subscribe service, so as to obtain secure event types. The name of an event type contains the public key of the type owner, so as to allow any entity to verify the integrity of an event type definition and to guarantee secure type and attribute names. Also in case of topics, it is possible to insert the public key of the owner in the name so as to make also the topic name secure. In [158], such a solution is further improved by considering delegation certificates to facilitate the Internet-scale management of event types. If a change to an event type is needed, its owner has to resign it. To avoid this, type managers have been introduced so as to make changes to event types on behalf of their owners. Delegation certificates define a digitally-signed path of trust from the event type owner to its type managers.

As above mentioned, a digital signature has to be coupled with a proper freshness scheme, so as to be able to provide non-repudiation and to tolerate replay attacks. From Table VIII, we can notice that not all the solutions that adopt a digital signature scheme exhibit also a freshness scheme. The majority of the solutions supporting freshness adopts timestamps, and only one uses nonces.

C. Secure Routing

Secure routing requires two main building blocks: (*i*) the secure maintenance of the broker state, and (*ii*) the secure notification forwarding. In the first case, we have to guarantee that, despite a possible attack, the service can continue to provide connectivity to all of its applications through the broker overlay. In the second case, event notification has to be

guaranteed despite possible malicious behaviors of brokers and applications.

To deal with the first aspect, a proper replication mechanism of brokers is needed. A preliminary work described in [98], where broker replication is proposed to provide Byzantine fault-tolerance in the event notification. This is still a matter of building replicated propagation paths and masking Byzantine faulty brokers. Both these mentioned works mask the effect of Byzantine brokers, but they do not exclude them from the system. Moreover, they do not consider a way of dealing with Byzantine publishers and subscribers, which may elude the access control mechanisms (i.e., an authorized node starts to behave in a Byzantine manner since it is controlled somehow by an adversary). Reference [98] proposes that border brokers adopt a Byzantine fault detector to identify inappropriate behaviors on the part of the publishers and/or subscribers and to punish them in some manner. Such a process is widely used to provide secure routing in wireless networks, such as [159], and in group communication services, such as [160]. This should be properly investigated also within the context of publish/subscribe services, as suggested by [98].

To deal with the first aspect, the generally recognized solution is to use redundancy paths from data sources to their destinations so as to tolerate possible malicious message drops, corruptions or mis-routing. In [127] and [132], multi-path routing is used to deal with message dropping attacks. Specifically, more than one path to a certain destination is established and several copies of a notification are sent along the multiple paths so as to achieve with a high degree of probability the successful delivery of the notification with high probability.

D. Auditing

Auditing is an important activity able to provide security within a system. As a matter of fact, several regulations for security in different application domains impose the adoption of auditing within the systems so as to enforce the achievable security guarantees. As a representative example, let us consider the Health Insurance Portability and Accountability Act (HIPAA),[1] which specifies standards for electronic healthcare transactions and data exchange. It has issued a HIPAA Security Rule where it demands the implementation of procedures to regularly review the records on the activities of the information system, such as audit logs.

Auditing at the middleware level is as important as that performed by operative systems and applications, so as to provide fruitful data for the detection of tentative security violations to break the mechanisms used by the middleware to protect itself. However, within the current literature on secure publish/subscribe services, this is an aspect that has been neglected, since none of the available solutions have proposed a manner for auditing the security decisions taken by the service. In [125], there is a reference to an accounting service; however, such a service only provides a log of all transformations made by PSAS to convert a notification encrypted with the publisher's key into one encrypted with the subscriber's key. Such a log is used to bill subscribers based on their usage of the publish/subscribe service. An accounting service should contain more than a simple log documenting the system usage by users, such as security-relevant chronological records providing evidence of the sequence of activities that have compromised a specific functionality, or portion of the middleware.

In other middleware platforms, auditing has received more attention than in publish/subscribe services. For concrete examples, the specification for security mechanisms in the CORBA platforms [161], as well as JBoss AS7[2] or Oracle Fusion Middleware,[3] includes auditing as a means to support security. The lack of a proper formalization of auditing within publish/subscribe services raises a number of important issues: (*i*) there is no consistency and comprehension between the auditing solutions built by different organizations, (*ii*) auditing data are scattered across the large-scale infrastructure with no proper solution to integrate them and to apply correlation operators for their analysis, and (*iii*) IT organizations spend a considerable amount of time and resources building and maintaining any suitable auditing solution.

E. Summary and Research Directions

The first consideration related to current academic solutions is that a significant number of them (i.e., 55%) are mostly theoretical models with no concrete implementations, as depicted in Table V. This limits the possibilities of a quantitative assessment of these solutions and their effective adoption in real use cases and/or within available commercial products. Another observation to make from a consideration of Table IX is that none of the available academic solutions presents a holistic approach to secure event notification, by maintaining that all aspects of security in publish/subscribe services (i.e., those that provide all the security means expressed in Fig. 8) should be taken into account and seen as a whole. On the contrary, the available solutions tend to focus on certain aspects by neglecting the others. Moreover, even if we consider a given aspect, such as access control, we can find differences between the proposed solutions: some assume trusted brokers (by not adopting broker access control or enabling brokers to hold plaintexts of the exchanged notifications), and control the access by publishers and/or subscribers, while others also deal with the access control of brokers. As shown in Table IX, the aspects of secure routing and auditing need more investigation within the context of publish/subscribe services, while the widely investigated aspects of access control and encryption must be considered as far from completely resolved and require further study since they still present some open issues. Here, we indicate some possible directions where we believe the research on secure publish/subscribe services should lead within the context of each required security means. We have summarized such research directions in Table X.

1) Access Control: As shown in Table VI, the access control models applied to publish/subscribe services are mostly

[1] http://www.hhs.gov/ocr/privacy/

[2] middlewaremagic.com/jboss/?p=453

[3] Oracle Fusion Middleware Audit Framework, description available at http://docs.oracle.com/cd/E12839_01/core.1111/e10043/audintro.htm.

TABLE IX
MAIN SOLUTIONS FOR SECURE PUBLISH/SUBSCRIBE SERVICES

Solution	Year	Access Control	Encryption	Signature	Secure Routing	Audit
[113]	2001	No	Yes	No	No	No
[114]	2002	Yes	No	No	No	No
[115]	2003	Yes	Yes	No	No	No
[117]	2006					
[118]	2003	Yes	No	No	No	No
[123]	2004	Yes	Yes	No	No	No
[125]	2005	No	Yes	Yes	No	Yes
[120]	2006	Yes	Yes	Yes	No	No
[121]	2007					
[122]	2011					
[126]	2006	No	Yes	No	No	No
[127]	2006	Yes	No	No	Yes	No
[129]	2006	Yes	Yes	No	No	No
[130]	2007	No	Yes	Yes	No	No
[131]	2007	Yes	Yes	Yes	Yes	No
[132]	2011					
[133]	2008	No	Yes	Yes	No	No
[134]	2009	No	Yes	No	No	No
[135]	2010	No	Yes	No	No	No
[136]	2010	No	Yes	Yes	No	No
[137]	2010	No	Yes	No	No	No
[138]	2012	Yes	Yes	No	No	No
[139]	2012	No	No	No	Yes	No
[140]	2012	No	Yes	Yes	No	No
[141]	2013					
[142]	2013	Yes	No	Yes	No	No
[143]	2014	No	Yes	Yes	No	No

TABLE X
RESEARCH DIRECTIONS ON SECURE PUBLISH/SUBSCRIBE SERVICES

Security Means	Research Topics
Access Control	Integration of break-glass solutions
	Control Decisions based on risk assessment or user reputation
	Integration of data owner permissions
Authentication	Realization of multi-technology federated IdM for SSO
Encryption	Application of batching for rekeying operations
	Use of key derivation to avoid key exchange
	Realization of robust key management
Digital Signature	Use of certificateless signatures
	Application of group signatures
	Introduction of Proxy signatures
Secure Routing	Application of proper replication mechanisms
	Application of multi-Path routing
Auditing	Support to integrity and availability in audit services
	Design of Intrusion Detection
	Provision of Self-Healing
All	Security Efficiency Assessment

the classic ones, namely those described in Section II-C4. Such models put great emphasis on how to protect sensitive information and/or certain functionalities of publish/subscribe services by not making them available to unauthorized subscribers, so as to guarantee data confidentiality. However, this is obtained at the expense of availability, i.e., the notifications containing important data must not be denied to authorized subscribers, but promptly provided to designated users when needed. In fact, the above-mentioned models regulate access control by granting access permissions under precise conditions and circumstances. However, it is not possible to consider all the possible circumstances of use of a given publish/subscribe service, and the relative access control mechanism may have to deal with an exceptional, unforeseen or dynamically-changing situation, and to take the decision of granting or not an access without proper prior instructions. To ensure availability in such situations, access mechanisms in use within publish/subscribe services include ways to override the applied access rules [162]. However, this introduces the possibility of abuses, such as eavesdropping, altering and skimming threats, and compromises the provided confidentiality. Hence, access control proves to be a challenge and requires a trade-off between confidentiality and availability [163]. A possible research topic is the study of extending the access control models for publish/subscribe services with break-glass solutions [164] (e.g., to bypass access control in emergency cases), and the use of auditing to monitor the cases in which the access control has been bypassed.

All the models for access control in publish/subscribe services are usually static: access is granted based on policies or rules set by the system administrator when the user is registered and subsequently seldom changed. In dynamic and critical application domains, such as the healthcare or finance, a more flexible model is required, where access grants are given not only based on static policies and rules, but also on certain dynamic characteristics of the users, such as his/her past behavior. In fact, the occurrence of success identity or authentication credentials theft is not a rare event, and this may make the conventional static access control models ineffective in revoking grants to malicious users. Therefore, researchers should investigate how to apply more advanced models that are recently emerging so as to obtain less rigid access decisions by dynamically increasing or decreasing the access to certain resources based on risk assessment [165] or user reputation [166]. Specifically, access decisions are taken based on the risk associated to the decision to be taken and the reputation of the requesting user. A proper reliable and secure system is needed to compute user reputation by collecting positive reward inputs and negative penalty points due to the interaction of users with the publish/subscribe service. Then, the user reputation score is computed based on the previous score and the received inputs, and indicates the probability that future interactions of the user with the system will be correct and allowable. An access can be granted if the user is trustworthy, e.g., his/her reputation score is above a given threshold. In addition, the risk of permitting an access can also be determined based on the criticality of the resource to be accessed and the user reputation [165].

Several application domains, such as healthcare, strictly require that data disclosure and dissemination are possible only if the person to whom the data is related has explicitly expressed his/her consent. This is mandatory to further enforce data confidentiality within the information system. As a concrete example, in several countries, proper regulations, such as the EU Data Protection Directive [167] or the Title X of the Public Health Service Act-42 CFR 59.11 [168] in the USA legislature, compel healthcare providers to obtain patient permission to disclose his/her health data. The current research on this topic is investigating ways to integrate such consent and access control, and how to introduce this within the context of publish/subscribe services.

2) Authentication: Authentication is a key aspect to consider when access control is introduced within a publish/

subscribe service. However, the available solutions have made very few efforts to devise it properly. Only two of them, i.e., [121], [141], have dealt with the issue of federated authentication, and only one, i.e., [141], has proposed a detailed architecture for IdM. Nowadays, publish/subscribe services are used as integration means for connecting and interoperating systems under different administration domains. Therefore, it is not reasonable to adopt a centralized or isolated architecture for IdM. The issues of a federated IdM and SSO are, therefore, crucial and need to be properly resolved. The solution of [121] to use delegation in X.509 certificates is not optimal, due to the limitations found within the X.509 specification: vulnerability to Man-In-The-Middle attacks when delegation is used [169]. Specifically, if an adversary can compromise one of the delegated certification authorities, then the thief can sign arbitrary certificates and compromise the overall security of the infrastructure. Moreover, these two solutions are based on a proper IdM framework and both use the same authentication means. In a heterogeneous large-scale infrastructure, using a publish/subscribe service as the integration and communication means, imposing the use of the same authentication technology is not viable. The service should allow each organization to keep its own authentication technology, and different technologies to coexist and interoperate by realizing a multi-technology SSO. Therefore, we believe that a future research topic is to investigate the most suitable way of making technology-independent federated IdM. In the current literature, such a topic has been studied extensively within the context of service-oriented computing and Web Services in particular, with several protocols for federated IdM being proposed and standardized (SAML is a concrete example of such a protocol). The research challenge is to investigate if the existing protocols can be successfully used also in the context of publish/subscribe services, or whether new ones are needed.

3) Encryption: Regarding the issue of how to improve encryption in event notification, the possible research topic is not to find the latest algorithm and apply it, but to improve one of the well-known and well-documented algorithms because they are already well-studied. Most of the cryptographic schemes in Table II have been used within the current literature. Symmetric encryption is widely acknowledged as the most suitable scheme for encrypting data of large size, since asymmetric cryptographic schemes are much slower than the symmetric ones. In publish/subscribe services used within the context of critical infrastructures, such as the above-mentioned examples of CI, timely event notification is an indispensable requirement. Therefore, the possibility of having fast encryption and decryption is a positive reason to prefer symmetric schemes. However, their adoption within the context of publish/subscribe services has been limited due to the scalability issues raised by the necessary rekeying. In the literature an attempt to resolve this issue has been adopted in [129] by means of key trees; however, we have noticed that this approach is not compatible with other issues affecting the performance and reliability of the event notification. Therefore, the open issue would be to strengthen a symmetric scheme by improving the management of the keys and reducing the limitations of rekeying group keys.

A solution would be to apply batch rekeying [170], which can reduce costs and overheads, limit inefficiency, and alleviate the out-of-sync problem: a central key manager collects join and leave requests in a period of time; when the number of received requests reaches a certain size (namely a batch), it rekeys. Despite being able to reduce the drawbacks of a traditional rekeying scheme, such a solution is, however, affected by a severe security flaw: being vulnerable to collision attacks. Specifically, when a member leaves a group, it still holds the group key. Therefore, there may be a delay in an old member passing the key to the new one joining the group later on. Over the years, several batch rekeying solutions have been proposed, e.g., by removing the central key manager so as to have a more distributed management [171], by combining batch rekeying with tree-based group keys so as to further improve efficiency [172], by introducing collision resistance so as to preserve secrecy within the message delivery [173], or by using proper recovery means so as to guarantee that new keys are eventually delivered to all users [174].

A possible different solution to reduce the overheads of rekeying caused by membership changes may be to avoid the need to exchange new keys by applying a proper key derivation technique [175]: new keys can be derived from the old keys by the members, without any key agreement managed by a centralized key manager or distributed between the members; but the opposite, i.e., the old keys derived from the new ones should be computationally infeasible. This allows the performance of rekeying operations and makes them non vulnerable to collusion attacks. Such an approach has been investigated within the context of secure multicast [176], [177], and it would be interesting to investigate its adoption also within the context of publish/subscribe services.

During key agreement a series of faults or attacks can occur, which can prevent the protocol from reaching agreement or even restarting. This implies a considerable performance degradation or even security vulnerabilities. It is crucial to make key agreement robust, in the sense that the protocol reaches an agreement even in case of the occurrence of faults or attacks. However, the solutions we have found in the current literature on key management in publish/subscribe services do not offer any means to enforce robustness, and prove to be susceptible to faults and attacks. Some solutions for robust key agreement are available in [178], [179], and they should be investigated so as to be applied to provide robustness in key management for secure publish/subscribe services.

4) Digital Signature: The current solutions for digital signatures in publish/subscribe services exhibit three main problems: (*i*) they require the presence of a trusted-by-all certificate authority to guarantee the relation between used keys and user identity; (*ii*) the publisher's identity is exposed during signature verification; and (*iii*) a destination needs to know the public keys of all the interacting publishers.

The first problem should be approached by using identity-based crypto systems so as to avoid the need to include certificates within the signatures or to interact with a certification authority to obtain one. Since the seminal work in [180] that introduced certificate-less signatures, a series of papers, such as [181], [182], have been proposed so as to further improve this

primal scheme and to make it more secure by removing the key escrow problem or more efficient by removing bilinear pairing (whose computations are heavier than the ones in traditional schemes) and basing the signature on the widely known RSA signature scheme [41].

Second, the current solutions realize the so-called Source Authentication, i.e., the possibility for the destination to verify the exact identity of the publisher for the received notifications. This violates the spatial decoupling of the service, since the identity of the publisher needs to be explicit and the event dissemination is no longer anonymous. A possible research direction in this sense is to investigate new means to authenticate messages without leaking signer identity and compromising anonymity within event notification. One of these means would be *Group Signature* schemes [183], which exhibit the following three properties:

1) only members of a given group can sign the messages;
2) destinations can verify if the signature is valid, without disclosing the true identity of the message signer;
3) the signature can be "opened" so as to reveal the identity of the group member that has signed the message.

A first practical solution to realize group signatures has been presented in [184] by using dynamic accumulators. However, the inefficiency of the available schemes has limited the widespread adoption of these group signature schemes. This issue has been considered in [185] and [186] to reduce the signature length, revocation capability, and signature creation/verification time. A simplified version of group signature schemes has been proposed in [187] named the *Ring Signature* scheme, where the signature creation and verification process is not assigned to a manager, but directly performed by the interested applications. The absence of managers allows the achievement of greater efficiency.

Last, the need for subscribers to know the public keys of the signers reduces the scalability of the signature schemes. Such an issue should be handled by applying the so-called *Proxy Signature* schemes [188], as partially implemented in [132]. In proxy signatures, the duty to create signatures is delegated to a given user/entity, called a proxy signer. Such a delegation can be realized in four different ways:

1) Full delegation: the private key of the original signer is given to the proxy signer, so that the proxy signer has the same signing rights as the original signer.
2) Partial delegation: the original signer and the proxy have two distant private keys, and the proxy signer is not limited in the range of messages it can sign.
3) Delegation by warrant: the original signer indicates which messages the proxy signer is allowed to sign.
4) Partial delegation by warrant: a combination of the previous two.

The latter has attracted some interest due to the advantage of indicating under which conditions the delegation is valid.

The new challenge related to signing notifications is how to introduce such schemes in a publish/subscribe service, especially a content-based one, or how to combine them, such as in [189]. Last, the current solutions do not adopt all the signature schemes shown in Table III, and a possible progress beyond the current state of the art would be to apply any scheme, such as the one in [190], able to realize a highly-efficient probabilistic scheme.

5) Secure Routing: In the current literature related to publish/subscribe services, we can find several approaches for reliable routing (reviewed in [10]), but very few for secure routing. A possible research study would involve rigorously investigating the combined application of BFT replication and multi-path routing to realize secure routing, which is lacking in the current literature. Moreover, the insertion of coding techniques within the multi-path routing, such as in [191] and [192], could be studied to reduce its intrinsic overhead and to further improve its provided notification guarantees.

6) Auditing: A challenging research topic consists in proposing and implementing an audit system in a publish/subscribe service so as to use the stored information for forensic analysis. Such a system should consider a crucial requirement: the integrity of the audit data is indispensable for its usage in the context of forensic analysis [193]. This means that (*i*) the integrity of the audit data needs to be protected, and (*ii*) the availability of the audit services has to be guaranteed by adopting proper replication. Therefore, research should focus on investigating proper techniques for the integrity and availability of audit services.

Another possible research topic would be to use the security logs to promote the self-monitoring and healing of publish/subscribe services by detecting possible security threats and autonomously taking the necessary measures to recover from them or avoid their subsequent activation. To achieve this aim, a proper approach to processing such logs is crucial. Sources of frustration when undertaking this include the fact that audit services have to cope with a huge amount of data, which is highly fragmented in the case of a distributed system. In fact, an attack on a part of a publish/subscribe service, e.g., one of its brokers, can spread to other parts, causing multiple apparently-independent entries in the audit service. To properly process the logs and determine the cause of a security issue, it is necessary to group together dependent log entries. This is possible by applying data coalescence techniques. We can distinguish between time coalescence [194], based on the assumption that log entries due to the same cause are close in time; spatial coalescence [195], based on the consideration that relates entries written closely in time but on different nodes of the system under study; and content-based coalescence [196], based on the grouping of entries by looking at their specific contents. An investigation is needed to determine which is the most suitable technique and how to use it to determine the overall manifestation process for a successful attack.

After this, it is possible to allow the publish/subscribe service to recover from a security issue by taking proper reconfiguration actions without any strong human intervention for its recovery. In the literature, there are some solutions for the self-healing of publish/subscribe services, such as [197], [198], but they consider reliability-related issues. A possible challenging investigation would be to study similar approaches within the context of improving the achievable security degree.

7) Security Efficiency Assessment: As mentioned, publish/subscribe services are widely adopted within the context of

TABLE XI
MAIN STANDARDS FOR SECURE PUBLISH/SUBSCRIBE SERVICES

Standard	Issuer	Year	Kind	Subscription Type
CORBA-ES [13]	OMG	2004	ALM	Channel-based
CORBA-NS [204]	OMG	2004	ALM	Type- and Channel-based with Content-based Filtering
DDS [5]	OMG	2007	TLM	Type- and Topic-based with Content-based Filtering
JMS [6]	SUN	1999	ALM	Channel- and Topic-based with Content-based Filtering
AMQP [15]	AMQP Consortium	2006	ALM	Topic-based with Content-based Filtering
WSN [14]	OASIS	2006	ALM	Topic-based

numerous CIs. Several international organizations have specified regulations, e.g., [199], [200], related to the development, implementation, validation and maintenance of CIs, requiring the production of certain evidence to empirically demonstrate the achievable quality of the infrastructure and the fulfillment of several non-functional requirements. To achieve this aim, there is a demand for quantitative means to assess the quality of the CIs from the security perspective, where the adopted publish/subscribe service plays a key role. Despite being rich in proposals to enhance the provided security level, the current literature lacks research studies on how to properly assess the adopted security mechanisms. In fact, past benchmarking experiences in publish/subscribe services have focused more on performance and reliability assessment [199], [201], neglecting security assessment. Such a deficiency should be rectified by future research.

V. INDUSTRIAL PRODUCTS

Nowadays, the available commercial and open-source products for the implementation of a publish/subscribe service are based on a well-known standard. In the following section, we list the main ones, summarized in Table XI, and highlight their main characteristics with respect to how to support security when notifying events.

A. CORBA Platforms

CORBA is an OMG specification for a middleware allowing objects to make requests for methods offered by remote objects and allowing data to be exchanged between remote objects. The communication is conveyed by the ORB, which interconnects clients and servers. Within the context of CORBA, we can find two specifications for publish/subscribe services, issued by OMG. CORBA-ES [13] is the first and defines a channel-based solution with an infrastructure-based overlay composed of a centralized broker gluing together publishers and subscribers. Since CORBA-ES presents certain problems and limitations, a second specification was issued by OMG, named CORBA-NS [202]. CORBA-NS is an evolution of the previous standard by introducing distributed broker topologies by means of channel federations, supporting content-based subscriptions, and providing QoS support for the notification dissemination. With respect to security, neither specification introduces any indications on how to secure event notification. In fact, since event suppliers are still CORBA objects, these specifications do not introduce any new security means tailored to event notification; but they leverage another CORBA, called CORBA-SEC [203], on securing object references and communication. CORBA has the function of introducing security within a CORBA system by means of an object service. It does not specify any specific security mechanisms, but indicates a standard architecture and some interfaces to integrate existing mechanisms in a CORBA system. Such an architecture is structured in three different levels:

- Level 0: SSL is integrated within the ORB to protect the communications, and provide integrity and confidentiality by means of encryption. The public-key security in the CORBA security environment also supports digital signature algorithms.
- Level 1: security is transparently added to CORBA objects without such objects being aware of the added security features (no changes to the application logic of the objects are needed), which mainly consists in authentication, access control and auditing.
- Level 2: CORBA objects are aware that some security features, such as non-repudiation, are introduced and their application logic encompasses the use of such features.

The first two levels contain mechanisms implemented within the CORBA ORB for a secure invocation of remote methods (for this reason the application logic can be unaware of their use), while the last level hosts mechanisms implemented within the CORBA objects to be secured. Fig. 11 shows the difference between an insecure CORBA invocation and a secure one with CORBA-SEC. Specifically, the communication logic at the client and server side is encapsulated respectively with the stub and the skeleton, in accordance with the well known Proxy and Skeleton design patterns [16]. In addition, at the server side we can find a POA, responsible for serving several objects (through their skeleton) with the incoming requests. With a traditional invocation, requests, originated within the clients, are given to the ORB through the stub so as to be forwarded to the server. The server receives the request and answers by passing a response to the ORB, which is delivered to the client. These requests and responses are conveyed by a TCP channel, as illustrated in Fig. 11. When the first two levels of CORBA-SEC are used, a secure invocation is performed: (*i*) the TCP channel is secured with SSL, and (*ii*) requests and responses have to pass through the security mechanisms. From the server application point of view, there is no difference between a

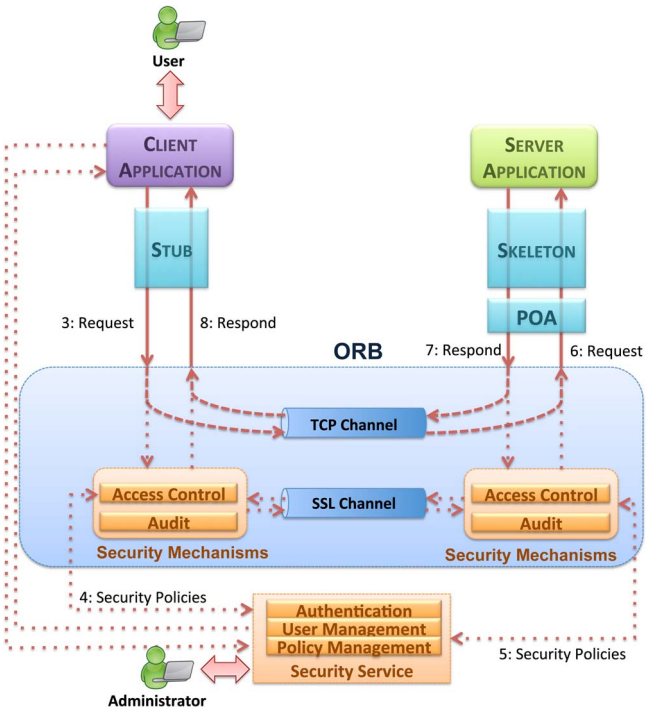

Fig. 11. Insecure (through TCP) and Secure Invocations with CORBA-SEC (through SSL).

traditional or a secure invocation, as clearly illustrated in the figure, since all the dirty work is performed by the ORB. Any user of a CORBA system, which can be a human operator or a software system, has a given *credential* associated, which collects a set of security *privileges* to access resources within a given CORBA system. A client application, therefore, has to first log in to the Security Service activated on the ORB. The login request has to contain a user name and password, or a certificate (previously registered at the Security Service); if successful, the response will be the credential of the user, which will be stored in the so-called *User* object. Such an object is accessed every time the clients try to perform a secure invocation. Specifically, before invoking a method on the target object for the first time, the stub makes use of the client-side access control and evaluates if it has the privileges to contact the target object. If this is the case, it sends the user credential to the target object. The server-side access control mechanism infers the privileges of such credential by also obtaining the security policies from the Security Service (as carried out also by the client-side access control mechanism). In the case of success, it establishes a security association with the client. Only after such an association has been successfully defined, is a secure invocation possible by exchanging data over the SSL channel, so as to ensure the integrity and confidentiality. CORBA-SEC does not specify how access control has to be implemented, leaving to the specific vendor the freedom to adopt any possible implementation. The specification only indicates the authorization model, with predefined access rights. In addition, a public-key cryptographic scheme can be also integrated within the ORB. Auditing is realized by logging all the operations performed during a secure invocation. The level 2 of CORBA-SEC is incomplete (i.e., it refers to other services and standards such as the ISO Non-Repudiation Framework [204]) and rarely implemented by current CORBA platforms.

B. DDS-Compliant Platforms

DDS [5] is another OMG standard, properly issued for publish/subscribe services, which, as mentioned, defines a topic- and type-based solution with content-based filtering based on TLM. The applications can exchange notifications thanks to a set of entities offered by the API of a DDS-compliant product [205]:

- A *Publisher* disseminates the notifications received from a publishing application.
- A *DataWriter* acts as a typed access to a publisher, i.e., it is specific for a given data structure (type). It is used by the application to pass to the Publisher the existence and value of data-objects with a given type. The association of a DataWriter with the Publisher expresses the intent of the application to publish the data described by the DataWriter in the context provided by the Publisher.
- A *Subscriber* receives the published data and makes them available to the receiving application.
- A *DataReader* behaves as a typed access attached to the subscriber. The association of a DataReader with the Subscriber expresses the intent of the application to subscribe to the data described by the DataReader in the context provided by the Subscriber.
- A *Topic* consists of a data type and a name, and connects a DataWriter with a DataReader. Specifically, events start flowing only when the Topic associated with a DataWriter matches the Topic associated with a DataReader.

The applications send and receive data within a *Domain*, a virtual space that partitions the overall communication infrastructure: only applications within the same domain can communicate. OMG also defined a standard "wire protocol" for DDS-compliant solutions, called DDS Interoperability (DDSI) [206], specifying that DDS-compliant solutions adopt IP Multicast. There is no standard specification related to security, at the moment. However, within the OMG there is a debate about this and a Request for Proposal for the security on DDS-compliant products[4] is available. To describe the solutions which the DDS community is moving towards, we have considered the two main DDS-compliant products and the mechanisms they provide for a secure event notification with DDS (shown in Fig. 12): RTI DDS[5] and OpenSplice DDS.[6] Both products exploit a secure transport for integrity and confidentiality. Specifically, RTI DDS wraps all the notifications to be exchanged using the Datagram Transport Layer Security protocol (DTLS) [207], which is a datagram based variant of SSL/TLS and supported by the open source OpenSSL library.[7] The DTLS protocol implies the secure authentication of the end points of a communication. To achieve this aim, it requires that: (*i*) there is a certification authority (known to all nodes interconnected by

[4]http://www.omg.org/members/cgi-bin/doc?dds/10-06-02.pdf
[5]http://www.rti.com/
[6]http://www.opensplice.com
[7]http://www.openssl.org/

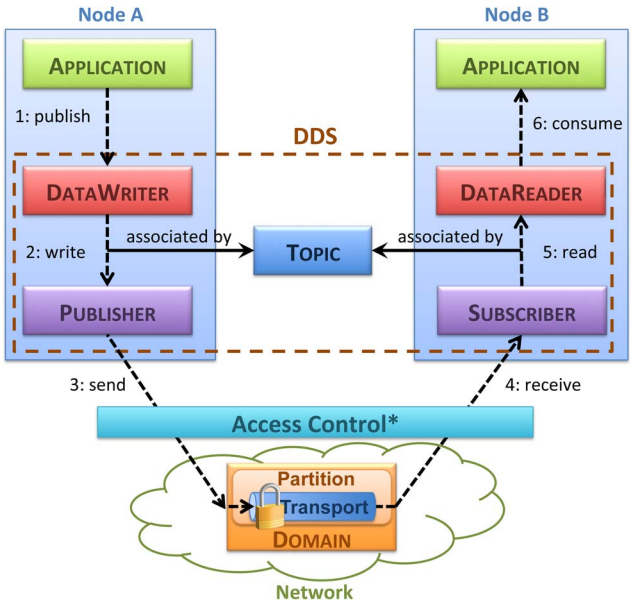

Fig. 12. Secure event notification for DDS-compliant middleware (*a proper Access Control only for OpenSplice DDS).

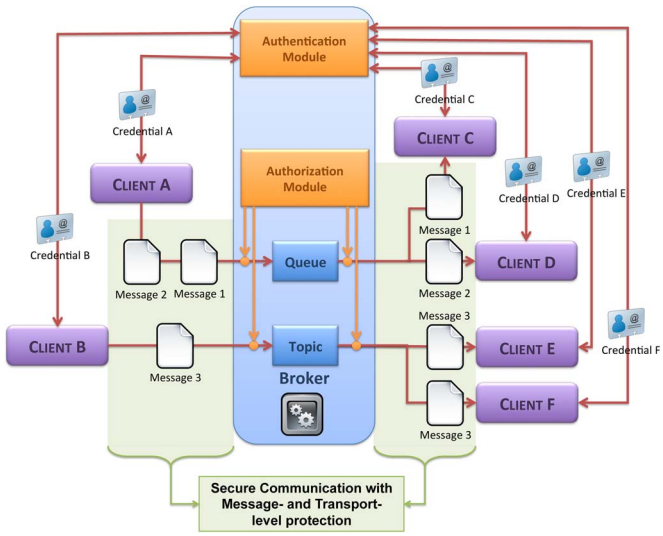

Fig. 13. Schematic Overview of security with JMS.

RTI DDS), (*ii*) each node has a certificate to identify it, which has been signed by the authority, and (*iii*) such a certification contains both the private and the public key used to encrypt and decrypt the notifications to be exchanged. DTLS does not have the ability to create multicast communication, since it has client/server semantics: there are a couple of unidirectional encrypted channels between two nodes, each playing the part of a client (when sending data) and a server (when receiving). Access Control is coarse-grain, achieved by partitioning a domain, so as to limit the visibility of the notifications to applications associated with certain partitions where the notification has been published. The programmer, in addition, is free to implement additional access control policies on the content of the notifications. On the other hand, OpenSplice DDS also partitions a domain, and each partition has associated a certain Security Profile for each Domain, which specifies the cipher and key to be used to encrypt the notifications exchanged over that domain. Its security module implements a variant of ACL:

- Each resource (object) is characterized by (*i*) a secrecy level, (*ii*) an integrity level, and (*iii*) a set of compartments that this resource is intended for.
- Each user (subject) has a clearance made by (*i*) a secrecy level, (*ii*) an integrity level, and (*iii*) a set of compartments he/she has a "need-to-know" for.
- To determine if a user is authorized to access a resource, e.g., he/she can publish notifications for a certain topic or subscribe to a given topic, the clearance of the user is compared to the security characterization of the resource:
  - A publication is allowed if the secrecy level of the resource is equal to or higher than that of the user;
  - A subscription is granted if the secrecy level of the resource is equal to or lower than that of the user;
  - A publication is allowed if the integrity level of the resource is equal to or lower than that of the user;
  - A subscription is granted if the integrity level of the resource is equal to or higher than that of the user;
  - A user can invoke the two operations if his/her compartments are a subset of those of the resource.

C. JMS-Compliant Platforms

JMS [6] is a very popular standard de facto for topic-based publish/subscribe services, which provides content-based filtering, and an infrastructure-based overlay with centralized and distributed broker topologies. Brokers hold proper containers for the notifications, which can be of two types, as shown in Fig. 13: *Queues* and *Topics*. In the first case, when a producer puts a notification in a queue, only one consumer, associated to this queue, receives the notification (realizing a point-to-point communication). On the contrary, when a producer puts a notification in a topic, all the consumers associated to this topic receive a copy of the notification. The standard does not specify which mechanisms the JMS providers must have to support security, which becomes very vendor-specific. So, the offered security options strongly depends on the adopted JMS provider. They are mainly composed of message integrity and confidentiality through encryption, and protection of JMS queues and topics by means of user authentication and authorization.

As a concrete example, let us consider the above-mentioned Apache Active MQ. Secure data transfers between clients and brokers are obtained by using a proper connector with message-level (with encryption and digital signatures) and transport-layer security (with SSL). Other security features are realized as a plug-in for the broker. Specifically, the broker is equipped with an authentication module, which is contacted every time a client has to establish a connection with the broker. Specifically, the client sends its credentials (in the form of user name/password or even of a certificate), as plain-text messages or as encrypted ones, to the broker, and if such a user has been defined in the broker authentication module, then the connection can be established. Built on top of the authentication, we find the authorization module, which allows defining fine-grained access control to the broker resources, e.g., only

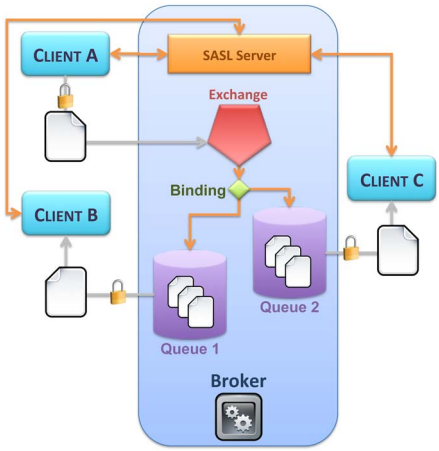

Fig. 14. Security in the AMQP specification.

specific authenticated and authorized clients can write to a given destination. Authorization can be at the operation level and at the message level. In the first case, for each destination, it is possible to state which users are allowed to perform a *read* (receive notification from the destination), *write* (send notification to the destination), or *administrative* operation (manage the destination). Therefore, in addition to what has been above described, when a client invokes an operation on the broker, its operation level privileges are checked by the authorization module. The operation is successfully performed only if the user is allowed; otherwise, an exception is raised. It is possible to adopt a message-level authorization, i.e., the authorization module allows or denies an operation, such as a read or a write, for a given message based on the message content. For a concrete example, it is possible to define an authorization policy for which certain messages can be consumed only by clients hosted in the same machine as the broker. Despite there being some plug-ins already available within Apache Active MQ that the user is able to use within his/her system, it is possible to define a custom plug-in so as to implement advanced authentication and authorization mechanisms. These security features apply not only among clients and brokers, but also among brokers within the overlay. If a given broker has the authentication enabled, then other brokers that wish to connect to that broker must provide the proper authentication credentials. Also among brokers, the authorization module controls which operations the authenticated brokers can try to perform and if they have the required privileges to perform them. Moreover, message-level and transport-layer security can be applied for communications among the brokers.

D. AMQP-Compliant Platforms

AMQP [15] is an open standard application layer protocol for message-oriented middleware rather than a standardized API such as JMS. It specifies a topic-based publish/subscribe service with content-based filtering, where the broker is composed of three key entities, as illustrated in Fig. 14:

- A *Queue* stores notifications until all the associated client applications have consumed them.
- An *Exchange* receives notifications from the applications.
- A *Binding* defines a relation between queues and exchanges so as to forward notifications received from certain exchanges to given queues by defining proper routing rules.

Therefore, a subscription is the creation/association of a queue, and the definition of a routing rule towards it, while the publication implies establishing an association to an exchange component. With respect to security, the AMQP specification indicates various network-level mechanisms organized in layers to define an authenticated and/or encrypted transport protocol conveying AMQP traffic, where one level can also be nested inside another. Specifically, one level is represented by the TLS protocol for encryption and the digital signature of exchanged data, while the other consists in the Simple Authentication and Security Layer (SASL) [208] framework for authentication. In fact, each interaction with the broker is established after an authentication at the SASL server.

E. WSN-Compliant Platforms

WSN [14] is a family of standards to support the introduction of topic-based publish/subscribe interaction scheme within the context of the Web Services, and is based on a infrastructure-less overlay between publishers and subscribers (in the so called WS-Base Notification specification). Moreover, it also provides a content-based filtering by using predicates evaluated over the XML message content of the message body. Last, it is able to realize an infrastructure-based overlay as defined in another standard called WS-Brokered Notification. Such a standard defines the interactions between the applications and the brokers, leaving to the implementers the freedom to structure the broker-to-broker interactions. A concrete example is provided by IBM WebSphere,[8] where a notification is disseminated as follows when a brokered topology is adopted:

- The Publisher creates notifications, and registers which topics will be published with a NotificationBroker (the topics are formalized according to the WS-Topics standard [14]). However, it does not have the responsibility of sending them to any interested consumer, which is allocated to the NotificationProducer that maintains a list of interested consumers and arranges for notification messages to be sent to those receivers. In the case of a brokered topology, the Notification Producer is part of the NotificationBroker, which implements its own strategy for notification delivery.
- The Subscriber sends the subscription requests to a NotificationBroker on behalf of a given Notification Consumer.
- The NotificationConsumer receives notifications and processes them according to their application logic. The specification describes the two notification schemes introduced. The default scheme is the push one, since the Subscriber registers with the Notification Broker not only a Subscription, but also the reference to the relative

[8]http://www-01.ibm.com/software/websphere/

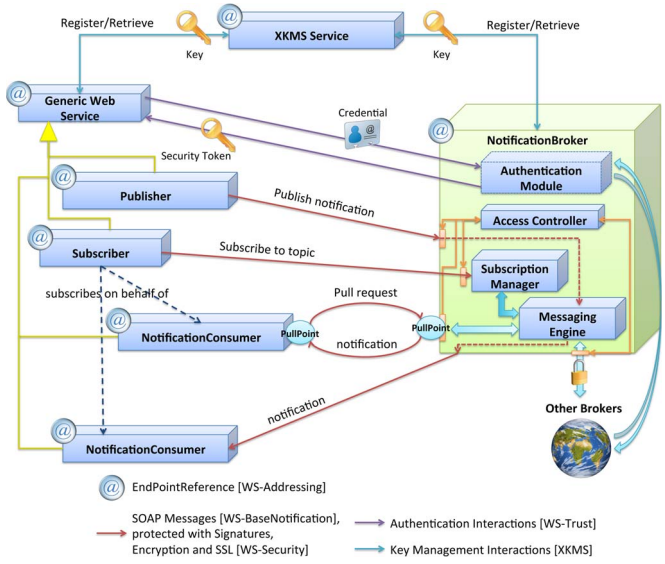

Fig. 15. Security among applications and brokers in WSN.

NotificationConsumer to be called as soon as a notification satisfying such a subscription is available. However, a pull-based notification is also available by defining a proper Pull Point, which is an entity that accumulates notifications and allows a requester to retrieve them.

- The Notification Broker is an intermediary Web Service decoupling NotificationConsumers from Publishers. Its main components are the Subscription Manager, to store subscriptions, and the Messaging Engine, to route incoming notifications towards other brokers.

Each of these entities has a remote reference specified according to a certain standard, i.e., WS-Addressing, and the exchanged SOAP messages have been formally encoded with a precise syntactical and semantic model of their composing fields. The standard itself does not specify any security mechanisms, since QoS policies like Reliability and Security can be offered by using a respective WS-* specification. In particular, a secure event notification with WSN is possible with the integration of the WS-Security specification [209], [210], as shown in Fig. 15.

WS-Security is a composite standard made by combining other different specifications and methods, and specifies two different levels of mechanisms to enforce the provided security level. The first is implemented at the message level by defining a SOAP header that carries out extensions to security. The second is realized at the service level to perform higher-level security mechanisms, such as access control or authentication. In particular, at the message level we can find two main XML security techniques with the function of protecting data exchange. The first is *XML Signature* [211]: a certain portion of the SOAP message to be exchanged is digitally signed (such an element is called a digest) so as to provide integrity and non-repudiation for the overall message content. The other one is *XML Encryption* [212]: a part of the overall SOAP message to be exchanged is encrypted by using a certain key, which can be public or private according to the chosen encryption strategy. Specifically, the SOAP header has a field, called DigestValue, to contain the digest with indications of the adopted signature method. When encryption is also used, the SOAP header has to contain the adopted encryption key, which is itself encrypted by using a proper public key. In addition to these two important message-level methods, SSL is also used. On top of these message-level mechanisms, we can find service-level mechanisms for authentication, authorization, and key management. SAML [79] is adopted to express authentication information in a request/respond manner when the communication participants do not share the same platform or belong to the same system. The core of this framework is the *assertion*, expressed as XML constructs, containing the identity of the requester, and the authorization decisions or credentials. Therefore, SAML conveys the result of an authorization process, with the assertion assumed to be a security token to access secured services, granted only by processing the information present in the assertion, which can be found in the SOAP header. SAML only deals with formalizing security tokens; how such tokens are obtained based on the provided credential is described in a proper specification named as WS-Trust.

Such a process is further enhanced with the above mentioned message-level mechanisms to achieve also confidentiality and integrity for the authentication interactions. Specifically, a given Web Service that intends to interact with a NotificationBroker has to acquire a security token by passing its identity to an authenticator, which can be within the NotificationBroker as in Fig. 15, or even an external Web Service. Such a security token is included in the header of every SOAP message forwarded to the NotificationBroker, and based on such a token an access control decision is taken. XACML [85] is used to specify roles and policies used by an access control mechanism to infer the access decisions for users. Within the overlay, we can have brokers belonging to different organizations, each with their own access roles and grants. XACML is used to exchange such decisions among brokers and to orchestrate their access decisions. In particular, it is possible that a broker returns to the user a security token that he/she can use also to access services and data hosted by another broker. XACML is used so that the access roles and policies of the first broker can be disseminated towards the second. Therefore, when the second broker receives a security token issued by the first, it is able to recognize it and to take the right access decision. eXtensible Rights Markup Language (XrML) is used to write rights and conditions related to the access control (such as expiration times). XML Key Management Specification (XKMS) [213] defines interfaces for the distribution of keys used in the message signature and encryption. For a concrete example, a given Web Service, named as Bob, registers its public key at an XKMS service. When another Web Service, named Alice, wants to send a message to Bob, it can interact with the XKMS service to obtain the public key of Bob. Such a key can be used to encrypt the SOAP message and send it to Bob, which can decrypt it with its private key. Within the WSN there can be more than one domain where Web Services are running and an XKMS service is available. The few XKMS services within the infrastructure cooperate with each other so that a key can be successfully retrieved on an XKMS service different from the one where the key has been registered. WS-Security does not contain any

TABLE XII
COMPARISON OF THE SECURITY GUARANTEES PROVIDED BY THE MAIN STANDARDS FOR PUBLISH/SUBSCRIBE SERVICES

Standard	Security Regulation	Access Control	Encryption	Signature	Secure Routing	Audit
CORBA-ES [13]	CORBA-SEC [205]	Yes	Yes	Yes	No	Yes
CORBA-NS [204]						
DDS [5]	Vendor-Specific	Yes	Yes	Yes	No	No
JMS [6]	Vendor-Specific	Yes	Yes	Yes	No	No
AMQP [15]	In the Standard	Yes	Yes	Yes	No	No
WSN [14]	WS-Security [211], [212]	Yes	Yes	Yes	No	Yes

indication on how to perform auditing; however, the literature of Web Services is full of solutions for the monitoring and logging of the interactions of clients with Web Services. Such approaches can be summarized in two different classes [214]. The most simple is to use the logging capabilities of the adopted platform running the Web Service to perform the monitoring. The other approach is to outsource the management of the Web Service to a third entity, especially in the case of composite Web Services, e.g., another Web Service [215] or an entity in a multi-agent system [216]. Such approaches can be used to realize security auditing.

F. Summary

Table XII summarizes our overview of the security assurances of commercial products based on proper standards. Such an overview highlights the fact that the marketed products do not apply the latest outcomes of academic research, and make use of solutions conceived within the context of unicast communications, but not tailored for event notification. This implies a loss of scalability and performance when security is enabled compared to a case with no security. Let us consider a concrete example of data encryption. In most commercial products, we have a link-by-link encryption. On the other hand, in most of the academic studies, we can find a different solution, referred to in this paper as end-to-end encryption.

VI. FINAL REMARKS

This paper has addressed the definition of a security model for publish/subscribe services by formally specifying related definitions and identifying threats that can compromise the degree of security offered. Moreover, it has also provided an analysis of current academic and industrial solutions in order to point out their pros and cons, and has identified what has been overlooked in past research activities and needs to be considered in future studies. The paper has also indicated what the possible directions of research in the near future could be.

Improving the security solutions for publish/subscribe services does not represent the only possible future research direction. Since there are several different possible solutions, there is a requirement to have proper means of evaluation to assess the pros and cons of the available alternatives and to decide which is the best. We believe this will represent a promising research topic in the future, since security evaluation is demanded whenever any given middleware is used to build critical systems whose development is regulated by proper standards.

REFERENCES

[1] P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surveys*, vol. 35, no. 2, pp. 114–131, Jun. 2003.

[2] C. Esposito, M. Ciampi, and G. De Pietro, "An event-based notification approach for the delivery of patient medical information," *Inf. Syst.*, vol. 39, pp. 22–44, Jan. 2014.

[3] M. Cinque, C. Di Martino, and C. Esposito, "On data dissemination for large-scale complex critical infrastructures," *Comput. Netw.*, vol. 56, no. 4, pp. 1215–1235, Mar. 2012.

[4] I. Delamer and J. Lastra, "Service-oriented architecture for distributed publish/subscribe middleware in electronics production," *IEEE Trans. Ind. Informat.*, vol. 2, no. 4, pp. 281–294, Nov. 2006.

[5] *Data Distribution Service (DDS) for Real-Time Systems*, OMG, Needham, MA, USA, 2007, Accessed: Sep. 2012. [Online]. Available: www.omg.org

[6] *Java Message Service*, Sun Microsystems, Santa Clara, CA, USA, 2002, Accessed: Sep. 2012. [Online]. Available: docs.sun.com/app/docs/doc/816-5904-10

[7] FuseSource, FuseSource Customers, Accessed: Jul. 2013. [Online]. Available: http://www.w3.org/TR/xmldsig-core

[8] Real-Time Innovations (RTI), Industries, Sunnyvale, CA, USA, Accessed: Apr. 2013. [Online]. Available: http://www.rti.com/industries/customers.html

[9] PrismTech Ltd., Industry Solutions, Tyne and Wear, U.K., Accessed: Apr. 2013. [Online]. Available: http://www.prismtech.com/opensplice/industry-solutions

[10] C. Esposito, D. Cotroneo, and S. Russo, "On reliability in publish/subscribe services," *Comput. Netw.*, vol. 57, no. 5, pp. 1318–1343, Apr. 2013.

[11] G. Muhl, L. Fiege, and P. Pietzuch, *Distributed Event-Based Systems*. Berlin, Germany: Springer-Verlag, 2006.

[12] T. Faison, *Event-Based Programming—Taking Events to the Limit*. Berkeley, CA, USA: Apress, 2006.

[13] *CORBA Event Service Specification*, OMG, Needham, MA, USA, 2001, Accessed: May 2011. [Online]. Available: www.omg.org/

[14] *Web Services Notification (WSN)*, OASIS, Burlington, MA, USA, 2006, Accessed: Sep. 2012. [Online]. Available: www.oasis-open.org/

[15] *Advanced Message Queueing Protocol*, v1.0, 2011, Accessed: Sep. 2012. [Online]. Available: www.amqp.org

[16] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns*. Reading, MA, USA: Addison-Wesley, 1995.

[17] D. Riehle, "The event notification pattern—Integrating implicit invocation with object-orientation," *Theor. Pract. Obj. Syst.*, vol. 2, no. 1, pp. 43–52, 1996.

[18] D. Garlan and D. Notkin, "Formalizing design spaces: Implicit invocation mechanisms," in *Proc. 4th Int. Symp. VDM Europe Formal Softw. Develop.*, vol. 551, *Volume I, Lecture Notes in Computer Science*, 1991, pp. 31–44.

[19] D. Schmidt, M. Stal, H. Rohnert, and F. Buschmann, *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*. Hoboken, NJ, USA: Wiley, 2000.

[20] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture: A System of Patterns*. Hoboken, NJ, USA: Wiley, 1996.

[21] F. Buschmann, K. Henney, and D. Schmidt, *Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing*. Hoboken, NJ, USA: Wiley, 2007.

[22] B. Grone, "Conceptual patterns," in *Proc. 13th Annu. IEEE Int. Symp. Workshop Eng. Comput. Based Syst.*, 2006, pp. 241–246.

[23] T. Schlossnagle, *Scalable Internet Architectures*. Indianapolis, IN, USA: Sams, 2006.

[24] B. Oki, M. Pfluegl, A. Siegel, and D. Skeen, "The information bus: An architecture for extensible distributed systems," *ACM SIGOPS Oper. Syst. Rev.*, vol. 27, no. 5, pp. 58–68, Dec. 1993.

[25] P. Eugster, "Type-based publish/subscribe: Concepts and experiences," *ACM Trans. Programm. Lang. Syst.*, vol. 29, no. 1, pp. 1–50, Jan. 2007.

[26] D. Rosenblum and A. Wolf, "A design framework for internet-scale event observation and notification," *ACM SIGSOFT Softw. Eng. Notes*, vol. 22, no. 6, pp. 344–360, Nov. 1997.

[27] R. Baldoni, L. Querzoni, S. Tarkoma, and A. Virgillito, "Distributed event routing in publish/subscribe systems," in *Middleware for Network Eccentric and Mobile Applications*. Berlin, Germany: Springer-Verlag, 2009, pp. 219–244.

[28] A. Carzaniga, M. Papalini, and A. Wolf, "Content-based publish/subscribe networking and information-centric networking," in *Proc. ACM SIGCOMM Workshop Inf.-Centric Netw.*, Aug. 2011, pp. 56–61.

[29] J. Martins and S. Duarte, "Routing algorithms for content-based publish/subscribe systems," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 1, pp. 39–58, 2010.

[30] K. Obraczka, "Multicast transport protocols: A survey and taxonomy," *IEEE Commun. Mag.*, vol. 36, no. 1, pp. 94–102, Jan. 1998.

[31] C. K. Yeo, B. S. Lee, and M. H. Er, "A survey of application level multicast techniques," *Comput. Commun.*, vol. 27, no. 15, pp. 1547–1568, Sep. 2004.

[32] M. Hosseini, D. Tanvir, S. Shimohammadi , and N. D. Georganas, "A survey of application-layer multicast protocols," *IEEE Commun. Surveys Tuts.*, vol. 9, no. 3, pp. 58–74, 2007.

[33] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1456–1471, Oct. 2002.

[34] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Trans. Comput. Syst.*, vol. 8, no. 2, pp. 85–100, May 1990.

[35] S. Deering *et al.*, "The PIM architecture for wide-area multicast routing," *IEEE/ACM Trans. Netw.*, vol. 4, no. 2, pp. 784–803, Apr. 1997.

[36] J. Kurian and K. Sarac, "A survey on the design, applications, enhancements of application-layer overlay networks," *ACM Comput. Surveys*, vol. 43, no. 1, pp. 5:1–5:34, Nov. 2010.

[37] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or multiple-tree: A comparative study of live P2P streaming approaches," in *Proc. 26th IEEE Int. Conf. Comput. Commun.*, May 2007, pp. 1424–1432.

[38] G. V. Chockler, I. Keidar, and R. Vitenberg, "Group communication specifications: A comprehensive study," *ACM Comput. Surveys*, vol. 33, no. 4, pp. 427–469, Dec. 2001.

[39] A. Carzaniga, D. Rosenblum, and A. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Trans. Comput. Syst.*, vol. 19, no. 3, pp. 332–383, Aug. 2001.

[40] M. Castro, P. Drushel, A. Kermarec, and A. Rowstrom, "Scribe: A large-scale and decentralized application-level multicast infrastructure," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1489–1499, Oct. 2004.

[41] W. Stallings, *Network Security Essentials—Applications and Standards*, 4th ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2010.

[42] B. Schneier, *Applied Cryptography: Protocols, Algorithms, Source Code in C*, 2nd ed. Hoboken, NJ, USA: Wiley, 1996.

[43] W. M. Petullo, X. Zhang, J. A. Solworth, D. J. Bernstein, and T. Lange, "Minimalt: Minimal-latency networking through better security," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2013, pp. 425–438, Accessed: Jul. 2013. [Online]. Available: http://eprint.iacr.org/

[44] A. A. Hasib and A. Haque, "A comparative study of the performance and security issues of aes and rsa cryptography," in *Proc. 3rd Int. Conf. Convergence Hybrid Inf. Technol.*, Nov. 2008, vol. 2, pp. 505–510.

[45] S. Goldwasser and S. Micali, "Probabilistic encryption," *J. Comput. Syst. Sci.*, vol. 28, no. 2, pp. 270–299, Apr. 1984.

[46] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. 17th Int. Conf. Theory Appl. Cryptogr. Tech.*, 1999, pp. 223–238.

[47] C. Fontaine and F. Galand, "A survey of homomorphic encryption for nonspecialists," *EURASIP J. Inf. Security*, vol. 2007, no. 1, pp. 013801:1–013801:15, Jan. 2007.

[48] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology, Lecture Notes in Computer Science*, vol. 196. Berlin, Germany: Springer-Verlag, 1985, pp. 47–53.

[49] P. Barreto, H. Kim, B. Lynn, and M. Scott, "Efficient algorithms for pairing-based cryptosystems," in *Proc. 22nd Annu. Int. Cryptol. Conf. Adv. Cryptol.*, vol. 2442, *Lecture Notes in Computer Science*, 2002, pp. 354–369.

[50] F. Zhang, R. Safavi-Naini, and W. Susilo, "An efficient signature scheme from bilinear pairings and its applications," in *Proc. PKC*, vol. 2947, *Lecture Notes in Computer Science*, 2004, pp. 277–290.

[51] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Security*, 2006, pp. 89–98.

[52] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Security Privacy*, 2000, pp. 44–55.

[53] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st ACM Symp. Theory Comput.*, May/Jun. 2009, pp. 169–178.

[54] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," in *Proc. 22nd Annu. IFIP WG 11.3 Work. Conf. Data Appl. Security*, 2008, pp. 127–143.

[55] M. O'Brien and G. Weir, "Understanding digital certificates," in *Proc. 2nd Int. Conf. Cybercrime Forensics Educ. Training*, London, U.K., Sep. 2008, pp. 1–9.

[56] *ISO/IEC 13888-3:2009—Information Technology—Security Techniques—Non-Repudiation—Part 3: Mechanisms Using Asymmetric Techniques*, ISO/IEC JTC 1/SC 27, 2009, Accessed: Jul. 2014. [Online]. Available: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=44735

[57] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[58] R. Cramer and V. Shoup, "Signature schemes based on the strong RSA assumption," *ACM Trans. Inf. Syst. Security*, vol. 3, no. 3, pp. 161–185, Aug. 2000.

[59] T. E. Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Proc. CRYPTO 84 Adv. Cryptol.*, 1985, pp. 10–18.

[60] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 6th ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2013.

[61] M. J. Wiener, "Performance comparison of public-key cryptosystems," *RSA Crypto Bytes*, vol. 4, no. 1, pp. 1–5, 1998.

[62] R. Merkle, "A certified digital signature," in *Proc. Adv. Cryptol.*, 1989, pp. 218–238.

[63] S. Rohde, T. Eisenbarth, E. Dahmen, J. Buchmann, and C. Paar, "Fast hash-based signatures on constrained devices," in *Smart Card Research and Advanced Applications*, vol. 5189, *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2008, pp. 104–117.

[64] G. Ateniese, D. Chou, B. de Medeiros, and G. Tsudik, "Sanitizable signatures," in *Proc. Eur. Symp. Res. Comput. Security*, vol. 3679, *Lecture Notes in Computer Science*, 2005, pp. 159–177.

[65] C. Brzuska *et al.*, "Security of sanitizable signatures revisited," in *Proc. 12th Int. Conf. Pract. Theory Public Key Cryptogr.*, Mar. 2009, pp. 317–336.

[66] M. Suzuki, T. Isshiki, and K. Tanaka, Sanitizable signature with secret information, Tokyo Inst. Technol., Tokyo, Japan, Accessed: Jul. 2014. [Online]. Available: http://www.is.titech.ac.jp/research/research-report/C/C-215.pdf

[67] H. Meijer and S. Akl, "Digital signature schemes for computer communication networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 11, no. 4, pp. 37–41, Oct. 1981.

[68] M. van Dijk, L. F. G. Sarmenta, C. W. O'Donnell, and S. Devadas, "Proof of freshness: How to efficiently use an online single secure clock to secure shared untrusted memory," MIT, Cambridge, MA, USA, 2006, Accessed: Jul. 2014. [Online]. Available: http://csg.csail.mit.edu/pubs/memos/Memo-496/memo496.pdf

[69] H. Kopetz and W. Ochsenreiter, "Clock synchronization in distributed real-time systems," *IEEE Trans. Comput.*, vol. C-36, no. 8, pp. 933–940, Aug. 1987.

[70] R. Baldoni, A. Corsaro, L. Querzoni, S. Scipioni, and S. Piergiovanni, "Coupling-based internal clock synchronization for large-scale dynamic distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 5, pp. 607–619, May 2010.

[71] S. Kremer, O. Markowitch, and J. Zhou, "An intensive survey of fair non-repudiation protocols," *Comput. Commun.*, vol. 25, no. 17, pp. 1606–1621, Nov. 2002.

[72] M. Mambo and E. Okamoto, "Proxy cryptosystems: Delegation of the power to decrypt ciphertexts," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. 80, no. 1, pp. 54–63, Jan. 1997.

[73] M. Jakobsson, "On quorum controlled asymmetric proxy re-encryption," in *Proc. 2nd Int. Workshop Pract. Theory Public Key Cryptogr.*, vol. 1560, *Lecture Notes in Computer Science*, 1999, pp. 112–121.

[74] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Trans. Inf. Syst. Security*, vol. 9, no. 1, pp. 1–30, Feb. 2006.

[75] A. Ivan and Y. Dodis, "Proxy cryptography revisited," in *Proc. Netw. Distrib. Syst. Security Symp.*, 2003, pp. 1–20.

[76] J. Torres, M. Nogueira, and G. Pujolle, "A survey on identity management for the future network," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 787–802, 2013.
[77] J. Lopez, R. Oppliger, and G. Pernul, "Authentication and Authorization Infrastructures (aais): A comparative survey," *Comput. Security*, vol. 23, no. 7, pp. 578–590, Oct. 2004.
[78] R. Housley, W. Polk, W. Ford, and D. Solo, Internet x.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, Accessed: Jul. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc3280.txt
[79] S. Cantor, J. Kemp, R. Philpott, and E. Maler, Assertions and Protocols for the Oasis Security Assertion Markup Language (SAML) V2.0-OASIS Standard, Accessed: Jul. 2013. [Online]. Available: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf
[80] Y. Cao and L. Yang, "A survey of identity management technology," in *Proc. IEEE Int. Conf. Inf. Theory Inf. Security*, Dec. 2010, pp. 287–293.
[81] *A Guide to Understanding Discretionary Access Control in Trusted Systems*, National Computer Security Center (NCSC), Jul. 2013. [Online]. Available: http://fas.org/irp/nsa/rainbow/tg003.htm
[82] D. Ferraiolo, D. Kuhn, and R. Chandramouli, *Role-Based Access Control*, 2nd ed. Norwood, MA, USA: Artech House, Jan. 2007, Print on Demand.
[83] T. Priebe, E. Fernandez, J. Mehlau, and G. Pernul, "A pattern system for access control," in *Proc. 18th Annu. IFIP WG 11.3 Work. Conf. Data Appl. Security*, Jul. 2004, pp. 235–249.
[84] L. Zhi, W. Jing, C. Xiao-su, and J. Lian-Xing, "Research on policy-based access control model," in *Proc. Int. Conf. Netw. Security, Wireless Commun. Trusted Comput.*, 2009, vol. 2, pp. 164–167.
[85] T. Moses, eXtensible Access Control Markup Language (XACML)-OASIS Standard, Accessed: Jul. 2013. [Online]. Available: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf
[86] S. Bistarelli, F. Martinelli, and F. Santini, "A formal framework for trust policy negotiation in autonomic systems: Abduction with soft constraints," in *Proc. 7th Int. Conf. Auton. Trusted Comput.*, 2010, pp. 268–282.
[87] Integrating the Healthcare Enterprise (IHE), Access Control, Accessed: Jul. 2013. [Online]. Available: http://www.ihe.net/Technical_Framework/upload/IHE_ITI_TF_WhitePaper_AccessControl_2009-09-28.pdf
[88] D. DeCouteau, M. Davis, and D. Staggs, Cross-Enterprise Security and Privacy Authorization (XSPA) Profile of XACML v2.0 for Healthcare Version 1.0-OASIS Standard, Accessed: Jul. 2013. [Online]. Available: http://docs.oasis-open.org/xacml/xspa/v1.0/xacml-xspa-1.0-cs02.html
[89] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulié, "Epidemic information dissemination in distributed systems," *IEEE Comput.*, vol. 37, no. 5, pp. 60–67, May 2004.
[90] P. Costa, M. Migliavacca, G. Picco, and G. Cugola, "Epidemic algorithms for reliable content-based publish-subscribe: An evaluation," in *Proc. 24th IEEE Int. Conf. Distrib. Comput. Syst.*, Mar. 2004, pp. 552–561.
[91] H. Li *et al.*, "Bar gossip," in *Proc. 7th Symp. Oper. Syst. Des. Implementation*, 2006, pp. 191–204.
[92] *Glossary of Key Information Security Terms*, National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, Apr. 2006. [Online]. Available: http://csrc.nist.gov/publications/
[93] *National Information Assurance Glossary*, Committee on National Security Systems, Fort Meade, MD, USA, Apr. 2010. [Online]. Available: http://www.cnss.gov/Assets/pdf/cnssi_4009.pdf
[94] I. Sommerville, *Software Engineering*. Reading, MA, USA: Addison-Wesley, 2004.
[95] E. Jonsson and T. Olovsson, "On the integration of security and dependability in computer systems," in *Proc. IASTED Int. Conf. Rel., Qual. Control Risk Assess.*, Nov. 1992, pp. 93–97.
[96] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 1, pp. 11–33, Jan.–Mar. 2004.
[97] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. Programm. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.
[98] C. Tiancheng and H. Meling, "Byzantine fault-tolerant publish/subscribe: A cloud computing infrastructure," in *Proc. 31st IEEE Symp. Rel. Distrib. Syst.*, 2012, pp. 454–456.
[99] M. Merideth *et al.*, "Thema: Byzantine-fault-tolerant middleware for web-service applications," in *Proc. 24th IEEE Symp. Rel. Distrib. Syst.*, 2005, pp. 131–140.
[100] C. Wang, A. Carzaniga, D. Evans, and A. Wolf, "Security issues and requirements for internet-scale publish-subscribe systems," in *Proc. 35th Annu. Hawaii Int. Conf. Syst. Sci.*, Jan. 2002, vol. 9, pp. 7–10.
[101] D. Gollmann, "Facets of security," in *Global Computing. Programming Environments, Languages, Security, Analysis of Systems*, vol. 2874, *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2003, pp. 192–202.
[102] J. Zhou and D. Gollmann, "Evidence and non-repudiation," *J. Netw. Comput. Appl.*, vol. 20, no. 3, pp. 267–281, Jul. 1997.
[103] K.-Y. Lam and D. Gollmann, "Freshness assurance of authentication protocols," in *Proc. Comput. Security–ESORICS*, vol. 648, *Lecture Notes in Computer Science*, 1992, pp. 261–271.
[104] J. Zhou and R. Deng, "On the validity of digital signatures," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 2, pp. 29–34, Apr. 2000.
[105] L. Hollaar and A. Asay, "Legal recognition of digital signatures," *IEEE Micro*, vol. 16, no. 3, pp. 44–45, Jul. 1996.
[106] A. Rowstrom and P. Drushel, "Pastry: Scalable, decentralized object localization and routing for large-scale peer-to-peer systems," in *Proc. IFIP/ACM Int. Conf. Distrib. Syst. Platforms*, vol. 2218, *Lecture Notes in Computer Science*, Nov. 2001, pp. 329–351.
[107] S. Banerjee and B. Bhattacharjee, "Scalable secure group communication over ip multicast," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1511–1527, Oct. 2002.
[108] R. Atkinson, "Toward a more secure internet," *IEEE Comput.*, vol. 30, no. 1, pp. 57–61, Jan. 1997.
[109] G. Urdaneta, G. Pierre, and M. V. Steen, "A survey of dht security techniques," *ACM Comput. Surveys*, vol. 43, no. 2, pp. 8:1–8:49, Feb. 2011.
[110] H.-A. Jacobsen *et al.*, "The PADRES publish/subscribe system," in *Handbook of Research on Advanced Distributed Event-Based Systems, Publish/Subscribe and Message Filtering Technologies*. Hershey, PA, USA: IGI Global, 2009.
[111] A. Wun, A. Cheung, and H.-A. Jacobsen, "A taxonomy for denial of service attacks in content-based publish/subscribe systems," in *Proc. Inaugural Int. Conf. Distrib. Event-Based Syst.*, Jun. 2007, pp. 116–127.
[112] P. Judge and M. Ammar, "Security issues and solutions in multicast content distribution: A survey," *IEEE Netw.*, vol. 17, no. 1, pp. 30–36, Jan./Feb. 2003.
[113] L. Opyrchal and A. Prakash, "Secure distribution of events in content-based publish subscribe systems," in *Proc. 10th USENIX Security Symp.*, Aug. 2001, pp. 1–16.
[114] Z. Miklos, "Towards an access control mechanism for wide-area publish/subscribe systems," in *Proc. 22nd Int. Conf. Distrib. Comput. Syst. Workshops*, 2002, pp. 516–521.
[115] Y. Yan *et al.*, "Implementing a prototype of the security framework for distributed brokering systems," in *Proc. Int. Conf. Security Manage.*, 2003, pp. 212–218.
[116] S. Pallickara and G. Fox, "Naradabrokering: A distributed middleware framework and architecture for enabling durable peer-to-peer grids," in *Proc. ACM/IFIP/USENIX Int. Conf. Middleware*, 2003, pp. 41–61.
[117] S. Pallickara *et al.*, "A framework for secure end-to-end delivery of messages in publish/subscribe systems," in *Proc. 7th IEEE/ACM Int. Conf. Grid Comput.*, 2006, pp. 215–222.
[118] A. Belokosztolszki, D. Eyers, P. Pietzuch, J. Bacon, and K. Moody, "Role-based access control for publish/subscribe middleware architectures," in *Proc. 2nd Int. Workshop Distrib. Event-Based Syst.*, 2003, pp. 1–8.
[119] P. Pietzuch and J. Bacon, "Hermes: A distributed event-based middleware architecture," in *Proc. 22nd Int. Conf. Distrib. Comput. Syst. Workshops*, 2002, pp. 611–618.
[120] L. Pesonen, D. Eyers, and J. Bacon, "A capability-based access control architecture for multi-domain publish/subscribe systems," in *Proc. Int. Symp. Appl. Internet*, 2007, pp. 222–228.
[121] L. Pesonen, D. Eyers, and J. Bacon, "Access control in decentralised publish/subscribe systems," *J. Netw.*, vol. 2, no. 2, pp. 57–67, Apr. 2006.
[122] B. Shand *et al.*, "Security policy and information sharing in distributed event-based systems," in *Reasoning in Event-Based Distributed Systems*, vol. 347. Berlin, Germany: Springer-Verlag, 2011, pp. 151–172.
[123] L. Fiege, A. Zeidler, A. Buchmann, R. K. Kehr, and G. Mühl, "Security aspects in publish/subscribe systems," in *Proc. 3rd Int. Workshop Distrib. Event-Based Syst.*, 2004, pp. 44–49.
[124] H. Parzyjegla, D. Graff, A. Schröter, J. Richling, and G. Mühl, "Design and implementation of the rebeca publish/subscribe middleware," in *From Active Data Management to Event-Based Systems and More*. Berlin, Germany: Springer-Verlag, 2010, pp. 124–140.
[125] H. Khurana, "Scalable security and accounting services for content-based publish/subscribe systems," in *Proc. ACM Symp. Appl. Comput.*, 2005, pp. 801–807.

[126] C. Raiciu and D. Rosenblum, "Enabling confidentiality in content-based publish/subscribe infrastructures," in *Proc. Securecomm Workshops*, 2006, pp. 1–11.

[127] Y. Zhao and D. Sturman, "Dynamic access control in a content-based publish/subscribe system with delivery guarantees," in *Proc. 26th IEEE Int. Conf. Distrib. Comput. Syst.*, 2006, pp. 1–8.

[128] R. Strom *et al.*, "Gryphon: An information flow based approach to message brokering," in *Proc. Int. Symp. Softw. Rel. Eng.*, Paderborn, Germany, 1998, pp. 1–2.

[129] G. Padmavathi and S. Annadurai, "A security framework for content-based publish-subscribe system," *J. Electron. Commerce Res. Appl.*, vol. 5, no. 1, pp. 78–90, Jul. 2006.

[130] A. Corman, P. Schachte, and V. Teague, "Quip: A protocol for securing content in peer-to-peer publish/subscribe overlay networks," in *Proc. 30th Australasian Conf. Comput. Sci.*, 2007, vol. 62, pp. 35–40.

[131] M. Srivatsa and L. Liu, "Secure event dissemination in publish-subscribe networks," in *Proc. 27th Int. Conf. Distrib. Comput. Syst.*, 2007, pp. 1–8.

[132] M. Srivatsa, L. Liu, and A. Iyengar, "Eventguard: A system architecture for securing publish-subscribe networks," *ACM Trans. Comput. Syst.*, vol. 29, no. 4, pp. 10:1–10:40, Dec. 2011.

[133] W. Wong, F. Verdi, and F. Magalhães, "A security plane for publish/subscribe based content oriented networks," in *Proc. ACM CoNEXT Conf.*, 2008, pp. 45:1–45:2.

[134] A. Shikfa, M. Onen, and R. Molva, "Privacy-preserving content-based publish/subscribe networks," in *Proc. Emerging Challenges Security, Privacy Trust—24th IFIP TC 11 Int. Inf. Security Conf.*, May 2009, pp. 270–282.

[135] W. Chen, J. Jiang, and N. Skocik, "On the privacy protection in publish/subscribe systems," in *Proc. IEEE Int. Conf. Wireless Commun., Netw. Inf. Security*, 2010, pp. 597–601.

[136] T.-H. Yuen, W. Susilo, and Y. Mu, "Towards a cryptographic treatment of publish/subscribe systems," in *Cryptology and Network Security*, vol. 6467, *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2010, pp. 201–220.

[137] S. Choi, G. Ghinita, and E. Bertino, "A privacy-enhancing content-based publish/subscribe system using scalar product preserving transformations," in *Proc. 21st Int. Conf. Database Expert Syst. Appl. I*, 2010, pp. 368–384.

[138] M. Ion, G. Russello, and B. Crispo, "Design and implementation of a confidentiality and access control solution for publish/subscribe systems," *Comput. Netw.*, vol. 56, no. 7, pp. 2014–2037, May 2012.

[139] M. Nabeel, N. Shang, and E. Bertino, "Efficient privacy preserving content based publish subscribe systems," in *Proc. 17th ACM Symp. Access Control Models Technol.*, Jun. 2012, pp. 133–144.

[140] M. Nabeel, S. Appel, E. Bertino, and A. Buchmann, "Privacy preserving context aware publish subscribe systems," in *Network and System Security*, vol. 7873, *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2013, pp. 465–478.

[141] A. Fongen and F. Mancini, "Identity management and integrity protection in publish-subscribe systems," in *Policies and Research in Identity Management*, vol. 396. Berlin, Germany: Springer-Verlag, 2013, pp. 68–82.

[142] M. Tariq, B. Koldehofe, and K. Rothermel, "Securing broker-less publish/subscribe systems using identity-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 518–528, Feb. 2014.

[143] L. Fiege, M. Mezini, G. Mühl, and A. Buchmann, "Engineering event-based systems with scopes," in *Proc. 16th Eur. Conf. Object-Oriented Programm.*, 2002, pp. 309–333.

[144] S. Farrell and R. Housley, An Internet Attribute Certificate Profile for Authorization, Accessed: Jul. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc3281.txt

[145] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Security Privacy*, 2007, pp. 321–334.

[146] J. Bacon, K. Moody, and W. Yao, "A model of oasis role-based access control and its support for active security," *ACM Trans. Inf. Syst. Security*, vol. 5, no. 4, pp. 492–540, Nov. 2002.

[147] A. Fongen, "Architecture patterns for a ubiquitous identity management system," in *Proc. 6th Int. Conf. Syst.*, Jan. 2011, pp. 66–71.

[148] A. Hegland, E. Winjum, and O.-E. Hedenstad, "A framework for authentication in nbd tactical ad hoc networks," *IEEE Commun. Mag.*, vol. 49, no. 10, pp. 64–71, Oct. 2011.

[149] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *ACM Comput. Surveys*, vol. 35, no. 3, pp. 309–329, Sep. 2003.

[150] E.-J. Goh, "Secure indexes," Cryptology ePrint Archive—Report 2003/216, 2003.

[151] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," *ACM Trans. Inf. Syst. Security*, vol. 7, no. 1, pp. 60–96, Feb. 2004.

[152] S. Mittra, "Iolus: A framework for scalable secure multicasting," *SIGCOMM Comput Commun. Rev.*, vol. 27, no. 4, pp. 277–288, Oct. 1997.

[153] C. K. Wong, M. G. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Trans. Netw.*, vol. 8, no. 1, pp. 16–30, Feb. 2000.

[154] W.-G. Tzeng and Z.-J. Tzeng, "A public-key traitor tracing scheme with revocation using dynamic shares," in *Public Key Cryptography*, vol. 1992, *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2001, pp. 207–224.

[155] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Proc. Int. Conf. Comput. Sci. Appl.*, vol. 5072, *Lecture Notes in Computer Science*, 2008, pp. 1249–1259.

[156] A. Pannetrat and R. Molva, "Multiple layer encryption for multicast groups," in *Proc. IFIP TC6/TC11 6th Joint Working Conf. Commun. Multimedia Security—Adv. Commun. Multimedia Security*, 2002, pp. 137–153.

[157] M. Bartel, J. Boyer, B. Fox, B. L. Macchia, and E. Simon, XML-Signature Syntax and Processing, Accessed: Apr. 2013. [Online]. Available: http://fusesource.com/community/fuse-customers/

[158] J. Bacon, D. Eyers, J. Singh, and P. Pietzuch, "Access control in publish/subscribe systems," in *Proc. 2nd Int. Conf. Distrib. Event-Based Syst.*, 2008, pp. 23–34.

[159] X. Su, G. Peng, and S. Chan, "Forbid: Cope with byzantine behaviors in wireless multi-path routing and forwarding," in *Proc. IEEE Global Telecommun. Conf.*, 2011, pp. 1–6.

[160] K. Kihlstrom, L. Moser, and P. Melliar-Smith, "The securering protocols for securing group communication," in *Proc. 31st Annu. Hawaii Int. Conf. Syst. Sci.*, 1998, vol. 3, pp. 317–326.

[161] F. Kon *et al.*, "Monitoring, security, dynamic configuration with the dynamic TAO reflective ORB," in *Proc. IFIP/ACM Int. Conf. Distrib. Syst. Platforms*, 2000, pp. 121–143.

[162] D. Povey, "Optimistic security: A new access control paradigm," in *Proc. Workshop New Security Paradigms*, 2000, pp. 40–45.

[163] M. Dekker and S. Etalle, "Audit-based access control for electronic health records," *Electron. Notes Theor. Comput. Sci.*, vol. 168, pp. 221–236, Feb. 2007.

[164] A. Brucker and H. Petritsch, "Extending access control models with break-glass," in *Proc. 14th ACM Symp. Access Control Models Technol.*, 2009, pp. 197–206.

[165] R. Shaikh, K. Adi, L. Logrippo, and S. Mankovski, "Risk-based decision method for access control systems," in *Proc. 9th Annu. Int. Conf. Privacy, Security Trust*, 2011, pp. 189–192.

[166] L. Su, W. Wang, and A. Niu, "Reputation service and reputation based access control," in *Proc. Int. Conf. E-Prod. E-Serv. E-Entertain.*, 2010, pp. 1–4.

[167] *Directive 95/46/EC on the Protection of Individuals With Regard to the Processing of Personal Data and on the Free Movement of Such Data*, The European Parliament and the Council of the European Union, Brussels, Belgium, 1995. Accessed: Dec. 2013. [Online]. Available: eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:NOT

[168] *42 CFR Part 59-Attachment B*, U.S. Department of Health and Human Services, Washington, DC, USA, 2000, Accessed: Jan. 2014. [Online]. Available: http://www.hhs.gov/opa/title-x-family-planning/title-x-policies/program-guidelines/

[169] R. Perlman, "An overview of PKI trust models," *IEEE Netw.*, vol. 13, no. 6, pp. 38–43, Nov. 1999.

[170] X. Li, Y. R. Yang, M. Gouda, and S. Lam, "Batch rekeying for secure group communications," in *Proc. 10th Int. Conf. World Wide Web*, 2001, pp. 525–534.

[171] P. Lee, J.-S. Lui, and D. Yau, "Distributed collaborative key agreement and authentication protocols for dynamic peer groups," *IEEE/ACM Trans. Netw.*, vol. 14, no. 2, pp. 263–276, Apr. 2006.

[172] M. Li, Z. Feng, N. Zang, R. Graham, and F. Yao, "Approximately optimal trees for group key management with batch updates," *Theor. Comput. Sci.*, vol. 410, no. 11, pp. 1013–1021, Mar. 2009.

[173] B. Li, Y. Yang, Z. Lu, B. Yuan, and T. Long, "Secure distributed batch rekeying algorithm for dynamic group," in *Proc. IEEE 14th Int. Conf. Commun. Technol.*, Nov. 2012, pp. 664–667.

[174] X. Zhang, S. Lam, D.-Y. Lee, and Y. Yang, "Protocol design for scalable and reliable group rekeying," *IEEE/ACM Trans. Netw.*, vol. 11, no. 6, pp. 908–922, Dec. 2003.

[175] Y. Dodis, R. Gennaro, J. Håstad, H. Krawczyk, and T. Rabin, "Randomness extraction and key derivation using the CBC, cascade and HMAC modes," in *Proc. Adv. CRYPTO*, vol. 3152, *Lecture Notes in Computer Science*, 2004, pp. 494–510.

[176] A. Pour, K. Kumekawa, T. Kato, and S. Itoh, "A hierarchical group key management scheme for secure multicast increasing efficiency of key distribution in leave operation," *Comput. Netw.*, vol. 51, no. 17, pp. 4727–4743, Dec. 2007.
[177] J.-C. Lin, K.-H. Huang, F. Lai, and H.-C. Lee, "Secure and efficient group key management with shared key derivation," *Comput. Std. Interfaces*, vol. 31, no. 1, pp. 192–208, Jan. 2009.
[178] S. Jarecki, K. Jihye, and G. Tsudik, "Flexible robust group key agreement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 879–886, May 2011.
[179] T. Rams and P. Pacyna, "A survey of group key distribution schemes with self-healing property," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 820–842, 2013.
[180] S. Al-Riyami and K. Paterson, "Certificateless public key cryptography," in *Proc. Adv. ASIACRYPT*, vol. 2894, *Lecture Notes in Computer Science*, 2003, pp. 452–473.
[181] R. Tso, X. Huang, and W. Susilo, "Strongly secure certificateless short signatures," *J. Syst. Softw.*, vol. 85, no. 6, pp. 1409–1417, Jun. 2012.
[182] J. Zhang and J. Mao, "An efficient RSA-based certificateless signature scheme," *J. Syst. Softw.*, vol. 85, no. 3, pp. 638–642, Mar. 2012.
[183] D. Chaum and E. Heyst, "Group signatures," in *Proc. Adv. EUROCRYPT*, vol. 547, *Lecture Notes in Computer Science*, 1991, pp. 257–265.
[184] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Proc. Adv. CRYPTO*, vol. 2442, *Lecture Notes in Computer Science*, 2002, pp. 61–76.
[185] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Proc. Adv. CRYPTO*, vol. 3152, *Lecture Notes in Computer Science*, 2004, pp. 41–55.
[186] S. Zhou and D. Lin, "Group signatures with reduced bandwidth," *Proc. Inst. Elect. Eng.—Inf. Security*, vol. 153, no. 4, pp. 146–152, Dec. 2006.
[187] R. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *Proc. Adv. ASIACRYPT*, vol. 2248, *Lecture Notes in Computer Science*, 2001, pp. 552–565.
[188] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures for delegating signing operation," in *Proc. 3rd ACM Conf. Comput. Commun. Security*, 1996, pp. 48–57.
[189] W. Lei and L. Daxing, "An efficient id-based proxy ring signature scheme," in *Proc. WRI Int. Conf. Commun. Mobile Comput.*, Jan. 2009, vol. 3, pp. 560–563.
[190] D. Johnson and A. Menezes, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," Univ. Waterloo, Waterloo, ON, Canada, 2000, Accessed: Mar. 2014. [Online]. Available: http://cacr.uwaterloo.ca/techreports/1999/corr99-34.pdf
[191] T. Cui, L. Chen, and T. Ho, "Optimization based rate control for multicast with network coding: A multipath formulation," in *Proc. 46th IEEE Conf. Decision Control*, Dec. 2007, pp. 6041–6046.
[192] Y. Yang, C. Zhong, Y. Sun, and J. Yang, "Network coding based reliable disjoint and braided multipath routing for sensor networks," *J. Netw. Comput. Appl.*, vol. 33, no. 4, pp. 422–432, Jul. 2010.
[193] M. Afzaal, C. D. Sarno, L. Coppolino, S. D'Antonio, and L. Romano, "A resilient architecture for forensic storage of events in critical infrastructures," in *Proc. IEEE 14th Int. Symp. High-Assur. Syst. Eng.*, 2012, pp. 48–55.
[194] J. Hansen and D. Siewiorek, "Models for time coalescence in event logs," in *Proc. Int. Symp. Fault-Tolerant Comput.*, 1992, pp. 221–227.
[195] D. Tang and R. Iyer, "Analysis and modeling of correlated failures in multicomputer systems," *IEEE Trans. Comput.*, vol. 41, no. 5, pp. 567–577, May 1992.
[196] A. Pecchia, D. Cotroneo, Z. Kalbarczyk, and R. Iyer, "Improving log-based field failure data analysis of multi-node computing systems," in *Proc. IEEE/IFIP 41st Int. Conf. Dependable Syst. Netw.*, Jun. 2011, pp. 97–108.
[197] R. Baldoni, R. Beraldi, L. Querzoni, and A. Virgillito, "A self-organizing crash-resilient topology management system for content-based publish/subscribe," in *Proc. Int. Workshop Distrib. Event-Based Syst.*, Jan. 2004, pp. 3–87.
[198] E. D. Nitto, D. Dubois, and R. Mirandola, "On exploiting decentralized bio-inspired self-organization algorithms to develop real systems," in *Proc. Int. Workshop Softw. Eng. Adapt. Self-Managing Syst.*, May 2009, pp. 68–75.
[199] C. Esposito, D. Cotroneo, and N. Silva, "Investigation on safety-related standards for critical systems," in *Proc. 1st Int. Workshop Softw. Certification*, Nov. 2011, pp. 49–54.
[200] G. Uchenick, "Middleware for security and safety critical systems," in *Proc. Embedded Syst. Europe*, May 2006, pp. 24–26.
[201] C. Esposito, S. Russo, and D. Di Crescenzo, "Performance assessment of omg compliant data distribution middleware," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, Apr. 2008, pp. 1–8.
[202] OMG. CORBA Notification Service Specification, ver. 1.1. Accessed: May 2011. [Online]. Available: www.omg.org/
[203] OMG, CORBA Security Specifications. Accessed: December 2012. [Online]. Available: www.omg.org/
[204] *Information technology-Open Systems Interconnection-Security Frameworks in Open Systems-Part 4: Non-Repudiation Framework*, ISO/IEC Std. Ref. 10181-4, 1996.
[205] G. Castellote, "OMG data-distribution service: Architectural overview," in *Proc. 23rd Int. Conf. Distrib. Comput. Syst. Workshops*, May 2003, pp. 19–22.
[206] OMG, DDS Interoperability Protocol (DDSI), Needham, MA, USA, Accessed: Sep. 2012. [Online]. Available: www.omg.org
[207] N. Modadugu and E. Rescorla, "The design and implementation of datagram TLS," in *Proc. Netw. Distrib. Syst. Security Symp.*, San Diego, CA, USA, Feb. 2004, pp. 1–7.
[208] A. Melnikov and K. Zeilenga, Simple Authentication and Security Layer (SASL), Jun. 2006. [Online]. Available: http://tools.ietf.org/html/rfc4422
[209] N. A. Nordbotten, "XML and web services security standards," *IEEE Commun. Surveys Tuts.*, vol. 36, no. 4, pp. 96–98, Apr. 2003.
[210] M. Naedele, "Standards for XML and web services security," *IEEE Comput. Mag.*, vol. 11, no. 3, pp. 4–21, 2009.
[211] D. Eastlake *et al.*, XML Signature Syntax and Processing (Second Edition)-W3C Recommendation, Accessed: Jul. 2013. [Online]. Available: http://www.w3.org/TR/xmldsig-core/
[212] D. Eastlake and J. Reagle, XML Encryption Syntax and Processing (Second Edition)-W3C Recommendation, Accessed: Jul. 2013. [Online]. Available: http://www.w3.org/TR/xmlenc-core/
[213] P. Hallam-Baker and S. H. Mysore, XML Key Management Specification (XKMS 2.0)-W3C Recommendation, Accessed: Jul. 2013. [Online]. Available: http://www.w3.org/TR/xkms/
[214] A. Tsalgatidou and T. Pilioura, "An overview of standards and related technology in web services," *Distrib. Parallel Databases*, vol. 12, no. 1/2, pp. 135–162, Sep./Nov. 2002.
[215] A. Chuvakin and G. Peterson, "Logging in the age of web services," *IEEE Security Privacy*, vol. 7, no. 3, pp. 82–85, Jun. 2009.
[216] C. Chandersekaran and W. Simpson, "An agent based monitoring system for web services," in *Proc. 16th Int. Command Control Res. Technol. Symp.*, Apr. 2011, pp. 84–89.

Christian Esposito received the degree in computer engineering from University of Naples Federico II, Napoli, Italy, in 2006, and received the Ph.D. degree at the same university, in 2009. He is a fixed-term Researcher at the Institute for High Performance Computing and Networking (ICAR) of the Italian National Research Council (CNR). His main interests include positioning systems for mobile ad-hoc networks, benchmarking aspects of publish/subscribe services, and reliability strategies for data dissemination in large-scale critical systems. He regularly serves as a reviewer in several leading journals and conferences in the field of Distributed and Dependable Systems.

Mario Ciampi received the M.Eng. degree in computer engineering from the University of Naples Federico II, Napoli, Italy, in 2004, a degree in European Master on Critical Networked Systems in 2007, and the Ph.D. degree in information engineering from the University of Naples Parthenope, Napoli, Italy, in 2011. He is a Technologist at the Institute for High Performance Computing and Networking (ICAR) of the Italian National Research Council (CNR). He investigates interoperable architectures, information systems, and distributed computing. He is Adjunct Professor at the University of Naples Federico II and member of iHealthlab, HL7 Italia, and @ITIM.